Factored Models for Neural Machine Translation

Pedro Dias Coelho
pedro.dias.coelho@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa, Portugal
January 2021

Abstract

Globalization has increased the importance of having high quality, quickly generated, and easily accessible translations. For the past few years, neural machine translation (NMT) has been the major chosen solution to fulfill this need. Among the vast research done in the field, novel ways of representing input data have been studied in an attempt to enrich the information contained in each word representation. One of them is factors, a mechanism where each word, instead of being represented by only the word itself, is defined by a bundle of features.

In our work, we test and evaluate the usability of the factors' code in Marian, an open-source NMT toolkit. We show how using factors impacts the quality of translations and the performance of this framework in terms of inference and training speed. We also contribute to its codebase by implementing concatenation as a method of combining the embeddings of the word and the factors.

We conduct three experiments, where we use factors for three different use cases and show how they improve the quality of the NMT systems' translations. We use them to inject terminology at run time, and show how combining word and factor embeddings by concatenating them is the most valuable option. We use them to represent subword splits, comparing two different methods previously proposed to do so. Finally, we use them to encode morphological information in an attempt to improve the translation quality of morphologically rich languages, showing promising results for the English-Romanian language pair.

1. Introduction

Globalization brings the need for communicating with people from all around the world, in any language, enhancing the importance of having high quality, quickly generated, and easy accessible translations. While human translators can produce high quality translations, they can be costly and time consuming. Machine Translation (MT) seeks to close this gap, by producing more cost-effective translations, in a more scalable manner.

The increasing availability of large amounts of data, and the advances done in the scope of Deep Learning [1], led to Neural Machine Translation (NMT), [2, 3] an approach to MT based on neural networks. Motivated by the desire to enrich the information contained in each word representation, Bilmes *et al*. [4] introduced the concept of **factors**, where each word, instead of being represented by only the word itself, is defined as a bundle of different features. Factors were first incorporated in NMT by Sennrich and Haddow [5], who applied factors to the source side, and by Wilken and Matusov [6], who used them for target data. Given the versatility of factors, these works used them to encode many kinds of different information, namely part-of-speech tags, morphological information, capitalization, as well as information about how a word is split into its subword units [7]. The usage of factors proved to be beneficial to improve different aspects of NMT systems, such as the generation of unknown words, the grammatical coherence of sentences, the disambiguation of homonyms, and decreasing decoding time, by reducing the size of the vocabulary for example. More recently, Dinu *et al*. [8] and Exel *et al*. [9] proposed factors as a solution to tackle the problem of domain adaptation by injecting custom terminology into neural machine translation at run time.

Accompanying all these developments, a variety of different NMT open-source toolkits have been developed. Among them, the Marian toolkit [10] stood out from the remaining ones, given its pure implementation in C++, its minimal dependencies, and its competitive benchmarks related to training and inference speed, as well as its cost-effectiveness [11]. However, some toolkits like Nematus [12], Sockeye [13] or Open-NMT [14] have already released software versions with the implementation of source and target factors, which has not happened with Marian until this date. Motivated by the reported positive results of using factors, the Connecting Europe Facility (CEF) decided to incorporate the inclusion of user-supplied factors as one of the milestones of their "User-Focused Marian" action, focused on improving Marian to address the needs

of CEF eTranslation, an automated translation tool from the EU, and to broaden its user base. It is in the scope of that project that this work has been developed.

Regarding the aeronautics industry, it has been reported that language barriers affect performance negatively, especially when it comes to maintenance [15, 16], as most of the maintenance personnel does not speak English as native language. Misunderstandings can lead to prolongations of the time it takes to restore an asset to operational readiness. Allowing maintenance technicians to communicate in their native language would aid in the technicians' training processes, as often training in English takes a part of it. Therefore, the new advances in machine translation, which have increased the speed and accuracy of automatic translation, while lowering the cost of having access to them, can help mitigate the reported problems caused by faulty communication.

Furthermore, factors in particular can have a key role in enriching the quality of these translations, not only by improving the performance of the system as a whole, but mainly due to their use for domain adaptation, allowing the injection of specific terminology to the generated translation, useful for the aeronautics industry as it is a field which has a considerable term base.

The main contributions of this work are the following:

- We extend the work from [8], by comparing different factor embedding options. We show that concatenation is the best option, and we also extended the experiment to three other language pairs;
- We compare two different approaches to represent the subword splits with factors ([5] and [17]). Also, we show that using factors for this use-case, despite resulting in positive results for English-German as reported in the literature, when extending it to other language pairs, the same positive behavior is not noticed;
- We used factors to improve the translation of morphologically rich languages, showing promising results for the English-Romanian language pair;
- We tested and evaluated the usability of the factors' code in the Marian toolkit. Furthermore, we contributed to the open source Marian's code base by means of the CEF's Marian action,[1] with the implementation of concatenation as an option to combine word and factor embeddings, as the original code only had sum as an available option.

This work is organized as follows: Section 2, covers the state of the art regarding factored neural machine translation (FNMT), detailing the changes that need to be done in an NMT architecture to incorporate factors. Our implementation of concatenation to combine word and factor embeddings is also explained, alongside the factor implementation in Marian. We then show three different applications for factors, the same three applications for which we used them in our experiments in Section 3, in which we report and analyze the results of the experiments that we carried out. Section 4 does an overview of the main achievements obtained with this work, leaving also some suggestions for future improvements.

[1] Code submission available in: https://github.com/marian-cef/marian-dev

2. Factored Neural Machine Translation

2.1. Factors

Factors can be defined as a mechanism by which we can represent words in a sentence, not only by the words themselves but as a bundle of features. With this, we can associate to each word several types of attributes as a way of enriching the information that each word contains when it is provided to the system. When representing a word with a set of F different features, the first feature ($k = 1$), is called lemma and it is either the original word or a simplified version of it, depending on what is represented in the remaining features $k \in \{2, ..., F\}$, which we call factors.

2.1.1 Source Factors

Sennrich and Haddow [5] changed a NMT architecture similar to [18] so that it would be able to process source factors. The main change was applied to the embedding layer. In the unaltered architecture, the word embeddings vector E_i of a word x_i in a sentence $x = (x_1, ..., x_N)$ is obtained using an embedding matrix U. In the factored architecture, each word is represented by a total of F different features, therefore, separate embedding vectors are computed for each one, which are then concatenated. Summarily:

$$E_i = \Big\Vert_{k=1}^{F} U_k^T x_{ik} \tag{1}$$

where $\|$ denotes vector concatenation, $U_k \in \mathbb{R}^{K_k \times m_k}$ is the kth feature embedding matrix, K_k is the vocabulary size of the kth feature, m_k is the embedding size of the kth feature, and $x_{ik} \in \mathbb{R}^{K_k}$ is an one-hot encoding vector indicating the value of the kth feature.

2.1.2 Target Factors

García-Martínez *et al.* [6] changed a NMT architecture similar to [18] to be able to handle target factors. First of all, the decoder embedding layer needed a factored embedding layer, similar to the one seen in Section 2.1.1. Furthermore, at the end of the decoder, the architecture was changed in

a way that instead of producing only one output (the predicted word), it predicted both the lemma and its factors in each time step. In the unaltered architecture, a predicted word $\hat{y}^{(t)}$ can be obtained by multiplying the last hidden state $h^{(t)}$ with the output embedding matrix V. If instead of a single word one wants to output a bundle of F features, then F different embedding matrices are needed, therefore predicting each feature $\hat{y}_k^{(t)}$ with:

$$\hat{y}_k^{(t)} = \text{softmax}(V_k^T h^{(t)} + c_k), \tag{2}$$

where $c_k \in \mathbb{R}^{K_k} \in$ and $V_k \in \mathbb{R}^{m \times K_k}$ are the kth feature bias and output embedding matrix respectively, and m the size of the last hidden state (in this case the same as the embedding size, m).

García-Martinez *et al.* [6] also found that applying a lemma dependency to the prediction of the factors improved the accuracy of the factor predictions and consequently the system results.

2.2. Factors in Marian

2.2.1 Source Factors

The implementation of source factors in Marian has some differences when compared to the one analyzed in section 2.1.1. Instead of having several embedding matrices for each one of the different features used to represent a word, Marian only has one large embedding matrix where all this information is encoded. Also, instead of concatenating the lemma and the factor embeddings, these are summed.

To embed a sentence x with N words, $x = (x_1, ..., x_N)$, where each word is represented by F different features, we start by creating a large embedding matrix $U \in \mathbb{R}^{K \times m}$, where K is the vocabulary size of all the features vocabularies K_k combined, $\sum_{k=1}^{F} K_k = K$. In fact, this matrix could be seen as the concatenation (by the rows) of the different factors embedding matrices $U_k \in \mathbb{R}^{K_k \times m_k}$ seen in section 2.1.1, with the constraint that they all have the same embedding dimension $m_k = m$.

When we process the input sentence, we start by constructing a sparse binary matrix $M \in \mathbb{R}^{N \times K}$. Each row of this matrix has a multi-hot encoded vector that encodes the information represented in each word of the sentence x. Marian concatenates all factor vocabularies into one, and so, we are able to encode in a single vector which are the lemma and factors represented in a given word, by means of multi-hot encoding. This matrix is then multiplied with the embedding matrix U, resulting in the word embeddings $E \in \mathbb{R}^{N \times m}$. With this multiplication, we end up summing the embeddings of the lemma and factors. A schematic representation of this process can be seen in Figure 1.

Even though this approach is beneficial from a computational point of view, as it limits the embedding process to a single matrix product, it has some drawbacks. Firstly, it forces that all the factors must have the same embedding dimension as the lemma. Secondly, this approach forces the factor embeddings to be summed with the lemma embeddings, not allowing other options such as concatenation. Thirdly, it does not favor tied embeddings [19] when only source or target factors are used, due to the fact that the same embedding matrix has to encode the lemma and the remaining factors, and so, if one side encodes both lemma and factors, while the other side only encodes lemmas, the size of the source and target matrices would not match and therefore they would not be eligible to be tied.

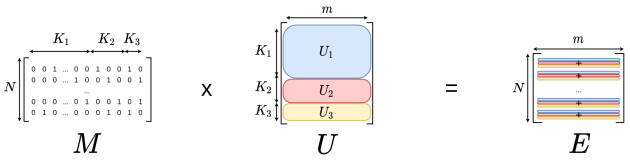

Figure 1. Representation of original lemma and factor embedding process of a sentence in Marian. Combining them with sum. For simplicity in this example a word is represented by a lemma and two factors ($F = 3$).

In an attempt to overcome these problems, we decided to implement a new method of combining lemma and factor embeddings, more similar to what we saw in section 2.1.1. To do so, we split the embedding process of the lemmas from the embedding process of the remaining factors. We created two embedding matrices $U_{lemma} \in \mathbb{R}^{K_1 \times m}$ and $U_{factors} \in \mathbb{R}^{(K-K_1) \times m_f}$, m_f being the embedding size of the factors, and K_1 the vocabulary size of the lemmas. To embed the lemmas, we can make use of the embedding matrix as a lookup table, and therefore avoid the matrix dot product, thus obtaining the lemma embeddings $E_{lemma} \in \mathbb{R}^{N \times m}$. Then for the remaining factors we create a multi-hot embedding matrix $M_{factors} \in \mathbb{R}^{N \times (K-K_1)}$ that is then multiplied with the factors embedding matrix $U_{factors}$, thus obtaining the factor embeddings $E_{factors} \in \mathbb{R}^{N \times m_f}$. Finally, the lemmas and factors embeddings are concatenated, in order to obtain the final embeddings $E = E_{lemma} \| E_{factors}$.

This approach favors the tying of the lemma embedding matrix of the source and target side even if only one of them uses factors. Furthermore, with this we are able to control the embedding size of the factors. A schematic representation of this process can be seen in Figure 2.

2.2.2 Target Factors

Regarding the target side, the implementation of the algorithm that predicts the different factors has also a different approach than the one explained in Section 2.1.2. There are four different factor prediction options in Marian (although one of them is only used for debugging purposes), that replace the last typical linear layer of the decoder archi-

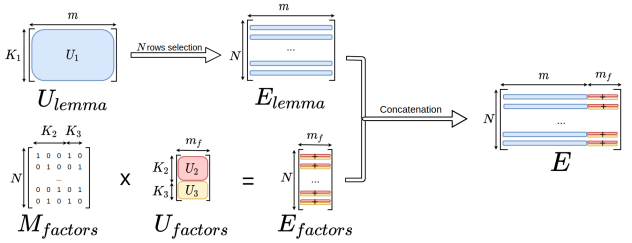

Figure 2. Representation of the implemented lemma and factor embedding process of a sentence in Marian. Combining them with concatenation. For simplicity in this example a word is represented by a lemma and two factors ($F = 3$).

tectures, all using a lemma dependency when predicting the factors. All these different methods start by obtaining the lemma prediction $\hat{y}_1^{(t)}$ by simply applying equation 2. This equation uses the output embedding matrix $V_1 \in \mathbb{R}^{m \times K_1}$, which is the output embedding matrix for feature $k = 1$, the lemmas. In Marian, as we did not have for the input embedding layer, we do not have several separate output embedding matrices V_k, but only one large output embedding matrix V. Although, as we mentioned in Section 2.2.1, and as it can also be seen schematically in Figure 1, the input embedding matrix U is in fact a concatenation by the rows of the sub embedding matrices U_k. Therefore, following the same pattern, in order to obtain V_1, we just need to select the rows from V correspondent to the first feature, and therefore, we obtain V_1, allowing us to use equation 2. The relevant factor prediction methods available in Marian are the following:

- **Soft Transformer Layer** - This name comes from the fact that the prediction of each factor is done with a condition mechanism that mimics a transformer layer [20]. After obtaining the lemma prediction $\hat{y}_1^{(t)}$, a multi-headed scaled dot product attention layer [20] is applied, receiving as inputs both the embedding of the predicted lemma and the outputted last hidden state by the decoder. Subsequently, as in a typical transformer layer, this is passed through a feed forward neural network (FFN), applying an "Add & Norm" layer after both the attention layer and the FFN. To the output of this condition mechanism $h_k^{(t)}$, equation 2 is applied, and the factor prediction $\hat{y}_k^{(t)}$ is obtained.

- **Lemma Custom Projection** - After obtaining the lemma prediction $\hat{y}_1^{(t)}$, this is projected to a new dimension m', with a new trainable parameter $D_1 \in \mathbb{R}^{K_1 \times m'}$, being reprojected again to dimension m by $D_2 \in \mathbb{R}^{m' \times m}$. Secondly, the lemma dependency is summed to the outputted hidden state instead of concatenated. Summarily:

$$\hat{y}_k^{(t)} = \text{softmax}\left(V_k^T \left(D_1 D_2 y_1^{(t)} + h^{(t)}\right) + c_k\right) \quad (3)$$

for $k \in \{2, ..., F\}$. If $m' = m$, only D_1 is used.

- **Lemma Dependent Bias** - to predict each factor with this method, a lemma dependent bias is computed for each factor, which is later added to the factor logits vector. The factor logits of the kth feature, is obtained with equation 2, but without the application of the softmax function. The lemma dependent bias b_k is obtained by multiplying the predicted lemma logits and a trainable parameter $B \in \mathbb{R}^{K_1 \times K_k}$. The factors are then obtained with,

$$\hat{y}_k^{(t)} = \text{softmax}(V_k^T h^{(t)} + c_k + b_k), \quad (4)$$

for $k \in \{2, ..., F\}$.

2.3. Factors Applications

2.3.1 Factors To Apply Terminology Constraints

Dinu *et al.* [8] proposed a successful method to inject custom terminology into neural machine translation at run time by using factors. The idea was to train the model in a way that it learned a "copying behavior", by providing the target term in the source side, enforcing the translation to produce that term. These target terms were integrated into the source sentences by appending them to their source version. The factors are used to signal this "code-switching" in the source sentence. Three factors were used (the numbers 0, 1, and 2), one to indicate source terms (1), another for the added target terms (2), and finally another for the words that were already part of the source sentence (0).

The lemma and the factor were embedded with a strategy similar to what we saw in Section 2.1.1, where lemma and factors have separate embedding matrices, and the resultant embeddings are concatenated. This was evaluated for the English-German language pair.

We extended this work by evaluating the impact of using factors to inject terminology at run time in other language pairs, and also compared different methods to combine the embeddings of the lemma and factors. This experiment is reported in Section 3.3.

2.3.2 Factors To Replace Subword Joining Markers

Both Sennrich and Haddow [5], and Wilken and Matusov [17], used factors to encode subword tags [7] information, following two distinct approaches:

- Sennrich and Haddow [5] used an annotation similar to the IOB format, having a factor that indicated if a certain subword unit was part of the beginning (B) of a word, the inside of a word (I), or the end of a word (E). (O) was used if a symbol corresponded to a full word, i.e, a word not split after applying BPE.

- Wilken and Matusov [17] used an annotation method that indicated when two consecutive subwords had to be merged. They used the factor "T" to indicate if a subword unit should be concatenated to the previous subword unit in a sentence, and used the factor "F" otherwise.

Both works tested this for the English-German language pair, reporting slight increases in BLEU score when comparing their systems to baselines.

In Section 3.4, we show how we extended their works, both by comparing directly these two encoding methods, but also by experimenting with other language pairs rather than English-German, to evaluate if using factors to encode subwords split information also has a positive impact on them.

2.3.3 Factors to Encode Morphological Information

García-Martínez *et al*. [6] deconstructed words into its morphological lemma and the respective morphological tags, both predicted by a morphological analyzer. The morphological tags were all concatenated into a string that was later treated as a single factor. Therefore, each word was represented by two features ($F = 2$), the lemma and the morphological tags factor.

After obtaining the predicted sequence of both the lemma and factors, for each pair of predictions, in post-processing, a morphological inflector is responsible for combining the lemma and its morphological information to generate the final word. New words not initially in the vocabulary, like a specific inflection of a certain word, could thus be generated. The authors reported slight increases in BLEU score for the English-German language pair when following this approach. Furthermore, the number of unknown words was reduced to more than half.

Motivated by the work done by García-Martínez *et al*. [6], we followed their approach of encoding morphological information into factors and tried to improve the translation of morphologically rich languages. These experiments are reported in Section 3.5.

3. Experimental Analysis

3.1. Datasets

In our experiments, we used four different language pairs: English-German (en-de), English-Latvian (en-lv), English-Norwegian (en-no) and English-Romanian (en-ro). For en-de, we used the data from the WMT 2018 English-German news translation shared task [21]. We gathered both the Europarl corpora [22] and the news commentary data corpora to form the training set. For the development (dev) and test sets, we used the 2013 and 2017 WMT English-German news translation shared task ([23], [24]) test sets respectively.

For the en-lv and en-ro language pairs, we joined the Europarl [22], DGT [25], JRC-Acquis [26], and EMEA corpora, all available through the OPUS data base [27], to create our datasets. For en-no, we used the Tilde-Model [28], data from the National Library of Norway, and finally data from the ELRC-SHARE repository. After collecting the data from these three language pairs, we removed empty and duplicate lines, as well as lines that were equal for the source and target side. We randomly selected 2.2 million sentences for the training data and 3000 for the dev set. Due to the smaller dimension of the en-no corpora, no random selection was needed, only the division between training and dev was done. Regarding test set, for en-lv we used the 2017 WMT English-Latvian news translation shared task [24] test set, and for en-ro the 2016 WMT English-Romanian news translation shared task [29] test set. For en-no, since it was never a language pair of WMT, when dividing the en-no corpora between train and dev sets, also 3000 random sentences were selected for the test set.

3.2. Baselines

3.2.1 Preprocessing Steps

After gathering the corpora for each language pair, we applied the same preprocessing sequence to all datasets. Firstly, we started by normalizing the punctuation and tokenizing the data. We subsequently applied truecasing, and finally, we applied BPE [7] to segment sentences into subword symbols, using 32000 merge operations, and the vocabulary threshold option set to 50. The BPE learned the merges over a concatenated corpus of the two languages in the language pair. Finally, we built shared vocabularies, in order to later tie the embeddings [19]. All the scripts used for the stages of the preprocessing pipeline prior to BPE were done with scripts from Moses [30].

3.2.2 Hyperparameters

The version of Marian used was 1.9.0. We trained Transformer models [20] with six encoder and decoder layers. The model dimension and consequently the embedding size was set to 512, the FFN size was set to 2048 with a depth value of 2, and we used 8 attention heads for the multi-head attention layers. Regarding batches, we used the Marian option of "batch-fitting", which determines the batch size automatically based on sentence-length to fit reserved memory (8500 Mb). We used dropout [31] of 0.1. Regarding optimization we used the Adam optimizer [32] ($\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$). Also, label-smoothing of 0.1 [33], a gradient clipping with a maximum gradient norm of 5 and exponential smoothing were applied. We limited the length of source sentences to 100 tokens. For decoding, we

used beam-search, with a beam size of 12. We used early-stopping so that training would stop when the model did not improve for 5 epochs. Finally, the source, target, and output embeddings matrices were tied [19]. All the baselines and FNMT architectures were trained under the same hyperparameter setup expect where noticed.

3.3. Factors To Apply Terminology Constraints

3.3.1 Experimental Setup

This experiment is an extension of the work done by Dinu *et al*. [8], analyzed in section 2.3.1. The goal of this experiment is to compare the method of combining lemma and factors embeddings, using our implementation of concatenation (Section 2.1.1), to the default Marian option to do so, which is combining them with sum (Section 2.1.1).

As seen in Section 2.3.1, Dinu *et al*. [8] added target terms to the source sentences, so that the model would learn to bias the translation to contain the provided terms. The target terms of the terminology were added to the input as inline annotations. We also use a factor group with three possible values ("p0", "p1", and "p2") to indicate if a word is either part of a source term ("p1"), part of a target term ("p2"), or if it is not part of a terminology pair ("p0"). The letter "p" is used due to the fact that in Marian all the factors of the same factor group must share the same prefix. The factors were added before the truecasing step. After applying BPE [7], if a word that was part of a term was split, the factors were shared among its subword units. In Marian, factors are placed directly in the corpus, by separating the lemma and the factors with a pipe character ("|").

Following the choices from [8], if a certain sentence has multiple matches from a term base, we keep the longest match. Furthermore, when checking for matches of a term inside a source sentence, we apply an "approximate matching", using a simple character sequence match, allowing a word in the text to be considered a match even if, for example, it happens to be inflected. Moreover, we also limit the amount of data added to 10% of the corpus as we want the model to work equally well when there are no terms provided as input.

Regarding the term bases used, for English-German, English-Latvian, and English-Romanian we used the European Union IATE term base, and for English-Norwegian we used two glossaries from eurotermbank. We then filtered out entries occurring in the top 500 most frequent English words as well as single character entries. The terminology used for the test and dev sets was terminology unseen during training.

For concatenation, we followed [8] and embedded the factors with a size of 16, and tied the source, target, and output lemma embedding matrices. As explained in Section 2.2.1, when we choose the option of combining the lemmas and factor embeddings with sum, they are embedded with the model embedding size. Also, tying the embeddings of the source and target side, when only one of them uses factors and when combining the lemma and factor embeddings by summing is, at first sight, difficult to do (Section 2.1.1). We worked around it by creating "dummy" factors for the target side that are actually never added to the target data, only to make the lemma and factor embeddings large matrix match in size, and therefore, tie the embeddings.

3.3.2 Results Analysis

By analyzing Table 1, starting by looking at the column that contains the results of the terminology percentage that was correctly translated, we can realize that the factored model outperforms the baseline for all language pairs. Furthermore, when comparing the two methods of combining lemma and factor embeddings, concatenation produces a slight increase (+0.5% to +2%) in the correctly translated terms percentage for all language pairs. Regarding the general quality of the translations, we can not take the same conclusion, as by analyzing the columns of the BLEU and COMET scores, the factored model that leads to the better results is not consistent for all the language pairs.

It is also noticeable that the COMET values for en-no differ a lot from the range of values for the other language pairs. This happens because the models for en-no generated some hallucinations in some of its translations. As COMET is not bounded unlike BLEU, these sentences are scored with greatly negative values, decreasing that way the average of the corpus score. Although, by looking at the leftmost column, we see that independently of that, the results for correctly translated terminology were positive.

Regarding tied embeddings, before realizing how to tie the lemma embeddings with the workaround explained in Section 3.3.1, we tried using lemma and factor embeddings combined with sum, without tying the source and target embeddings matrices. The results can be seen in Table 2.

By analyzing it, we see that the quality of the translations drops considerably (-3.13 BLEU and -0.1468 COMET) when comparing the untied and the tied model, what was expected. Regarding the percentage of correctly translated injected terminology, the untied model still slightly outperforms the baseline (+2%), although it underperforms (-5.8%) when compared to the tied model. With this, we can infer that tying the lemma embeddings is also important to guarantee a better performance of the information carried within the factors.

3.4. Factors To Replace Subword Joining Markers

3.4.1 Experimental Setup

In this experiment, we use factors to replace the subword [7] joining marker ("@@") and use factors both on the source and the target side.

		Term%	BLEU	COMET
en-de	Baseline	78.0	24.75	0.3918
	Factored model - sum	85.8	24.78	0.3863
	Factored model - concatenation	**86.3**	**24.86**	**0.4009**
en-lv	Baseline	56.8	15.90	0.3769
	Factored model - sum	71.2	**15.96**	0.3698
	Factored model - concatenation	**73.9**	15.67	**0.3830**
en-no	Baseline	83.1	22.04	**-0.4013**
	Factored model - sum	91.7	**22.17**	-0.4067
	Factored model - concatenation	**93.2**	22.00	-0.4145
en-ro	Baseline	81.8	24.13	0.4048
	Factored model - sum	92.9	**24.60**	**0.4065**
	Factored model - concatenation	**94.9**	24.29	0.3865

Table 1. Results of the experiment where factors are used for terminology injection.

	Term%	BLEU	COMET
NMT baseline	78.0	24.75	**0.3918**
Untied FNMT model	81.0	21.65	0.2395
Tied FNMT model	**85.8**	**24.78**	0.3863

Table 2. Results comparing tying and not tying the lemma embeddings. Both FNMT models used sum as the method to combine factor and lemma embeddings. Evaluated for the en-de language pair.

This experiment has the following objectives: Compare the different decoding options present in Marian (Section 2.2.2), in terms of performance and speed; compare two approaches taken in the literature ([5], [17]) to encode subword splits into factors; and evaluate if replacing BPE joint markers with factors improves the quality of the translations for the four language pairs explored in this work.

Regarding the comparison of the two methods, firstly, following [5] (Section 2.3.2) we also used a factor group with 4 possible values ("bb", "bm", "bf", and "bn"). The factor "bb" indicates if a certain unit in the text forms the beginning of a word, "bm" indicates the middle of a word, "bf" indicates the end of a word, and finally "bn" indicates that a certain unit is a full word.

For the other method, following [17] (Section 2.3.2) we use a factor group with 2 possible values ("jt", "jn"). These two factors indicate either if a certain token in a sentence should be joined with the previous token ("jt"), or if it should not be joined ("jn").

Before comparing the two approaches, we compared the different factor prediction options available in Marian and detailed in section 2.2.2. For the lemma custom projection, we tried three different lemma projection sizes: 16, 100, and 512. To compare this, we used the second method of encoding the BPE splits information, the one that uses the factors "jt" and "jn" (the one based in [17]).

We later used the option "softmax transformer layer" for comparing the two approaches to encode BPE splits information into factors, and to extend the experiment to the other language pairs. As concatenation was not implemented for the target side, the regular Marian factors embedding was used (Section 2.2.1), and all the embedding matrices were tied.

3.4.2 Results Analysis

By analyzing Table 3, where we have the comparison of the different Marian factor prediction options, we can see that if evaluating based on BLEU, the model that generated the best translations was the one that uses the soft transformer layer to predict the factors, while the model that had the highest COMET score was the model that reprojects the lemma embeddings to a layer with size 512. We can also see that the only model that underperformed the baseline in both BLEU and COMET is the one that predicts factors with the lemma custom projection of size 16, probably due to not capturing enough features given the short dimension of the lemma reembedding size.

In terms of speed, in the "Words/s" column of Table 3, the values reported are an average of the words processed per second during training. The rightmost column reports the time in seconds that it took to translate the en-de newstest set (3004 sentences) with a batch size of 16. This was trained and evaluated in a GeForce RTX 2080 Ti GPU. We can see that using target factors decreases the speed of both training and inference. All the methods to predict factors were slower than the baseline, which means that, even though the vocabulary size is reduced when replacing the BPE joint markers "@@" by factors (from 30915 tokens to 28653), it does not balance the increase in the number of trainable parameters and in the complexity of the model,

		BLEU	COMET	Words/s (Training)	Dec. Time [s]
en-de	Baseline	24.75	0.3918	**17197**	**50.35**
	Soft transformer layer	**24.89**	0.3919	13336	62.13
	Lemma Custom Projection (16)	24.59	0.3772	15797	56.94
	Lemma Custom Projection (100)	24.70	0.3924	15165	57.27
	Lemma Custom Projection (512)	24.78	**0.3981**	13530	61.23
	Lemma Dependent Bias	24.78	0.3821	15191	56.30

Table 3. Results on the comparison of the different factor prediction options available in Marian. The BPE splits factor encoding method is the one based on [17].

		BLEU	COMET
en-de	Baseline	24.75	0.3918
	Factored model (1)	24.85	0.3719
	Factored model (2)	**24.89**	**0.3919**
en-lv	Baseline	15.90	**0.3769**
	Factored model (1)	**15.97**	0.3442
	Factored model (2)	15.50	0.3707
en-no	Baseline	**22.04**	**-0.4013**
	Factored model (1)	21.10	-0.4451
	Factored model (2)	21.11	-0.4184
en-ro	Baseline	**24.13**	**0.4048**
	Factored model (1)	24.06	0.3531
	Factored model (2)	23.76	0.3960

Table 4. Results on encoding BPE splits as factors with two different methods, for different language pairs. (1) - BPE factor encoding based on [5]. (2) - BPE factor encoding based on [17].

thus not allowing for a decrease in training and translation times. The soft transformer layer and the lemma custom projection with a size of 512 are the slowest as they are the ones that introduce the highest number of trainable parameters. However, these also turn out to be the cases that generate the better translations, so these should be the choice when time is not an issue. Furthermore, the lemma dependent bias and the lemma custom projection with a medium size (100) appear to be the methods with the best balance between speed and quality of the translations.

Regarding the comparison of the two methods to encode BPE splits as factors when looking at Table 4, we can see that if compared with BLEU, the method proposed by Sennrich and Haddow [5] seems to be more effective for en-lv and en-ro that the method proposed by Wilken and Matusov [17]. However, when looking at the COMET column the same conclusion cannot be taken, as the method proposed by Wilken and Matusov [17] outperforms the method proposed by Sennrich and Haddow [5] for all the language pairs with a considerable margin (between +0.02 and +0.0429). Besides this, when comparing the highest scoring factored model with the baseline for each language pair, only for the en-de language pair we see a positive result for which the best scoring factored model (the one with the method proposed by Wilken and Matusov [17]), outperformed the baseline on BLEU and on COMET. Even though the increase on the latter is too small (+0.0001), at least we can conclude that the performance has not been compromised. For the remaining language pairs, only for the language pair en-lv we see a factored model outperform the baseline on BLEU (+0.07), although the same comparison by COMET (-0.0327) does not report the same positive result.

We can conclude that even though encoding BPE splits with factors seems to have a positive impact for English-German, as also reported in the literature [5, 17], when extending it to other languages where this was never tested, it does not lead to the same performance improvements. Nevertheless, the method proposed by Wilken and Matusov [17] proved to be more effective than the method proposed by Sennrich and Haddow [5].

3.5. Factors to Encode Morphological Information

3.5.1 Experimental Setup

In order to encode morphological information into factors, there are a total of three systems that should be used. Firstly, a morphological tagger [34] to predict the morphological tags of a certain word. Secondly, a lemmatizer [34] which reduces each word to its morphological lemma. And thirdly, a morphological inflector [35], that after the translation is predicted by the NMT model combines the lemma and the respective morphological tags and generates the final form of a certain word with the correct inflection. The models were trained under the Universal Dependencies (UD) Treebanks [36], although the UD schema was converted to the UniMorph schema [37] instead. This schema attempts to standardize the tags used to represent morphological features. Results on the accuracy obtained for these three models after they were trained can be seen in Table 5.

Regarding the factors used, we combined all the morphological tags into a single string so that they would be treated as a single factor. Furthermore, the word is replaced by its morphological lemma. For example, the word "does" would be represented as, "do|mV;FIN;IND;PRS;3;SG".

	Tagger	Lemmatizer	Inflector
German	82.34	96.17	91.93
Latvian	88.92	93.26	88.69
Norwegian	93.76	97.33	93.12
Romanian	94.84	96.68	91.62

Table 5. Accuracy obtained for the morphological tagger, lemmatizer and morphological inflector.

Furthermore, if a certain token does not carry morphological information, as punctuation characters for example, the factor "mNOFACT" was used. If factors are used for the target side, after predicting the lemma and the factors with the FNMT model, if a certain token is predicted with the "mNOFACT" factor, the predicted lemma is kept unchanged.

The factors with the morphological tags and the replacement of the words with their morphological lemmas was done before the truecasing step, for the non-English language of a given language pair. After applying BPE [7], the factors information was shared among the subword units of a split word.

Given the neural nature of the tagger and the lemmatizer, the time needed to preprocess the data increased dramatically. In order to decrease the preprocessing time, we reduced the training sets to half the size, therefore retraining also the baselines with the shorter datasets so that the comparison between the NMT model and FNMT model could be fair. Both directions of each language pair were used.

The lemma and factor embeddings were combined by summing them (Section 2.2.1), and when using them on the target side, the method used to predict the factors was the softmax transformer layer (Section 2.2.2). The source and target embedding matrices were tied using the workaround described in Section 3.3.1.

3.5.2 Results Analysis

		BLEU	COMET
de-en	Baseline	**27.66**	**0.3125**
	Factored model	27.19	0.2908
lv-en	Baseline	**12.87**	-0.1521
	Factored model	12.43	**-0.1492**
no-en	Baseline	**23.73**	**-0.5505**
	Factored model	23.00	-0.5562
ro-en	Baseline	23.77	0.0942
	Factored model	**26.65**	**0.2468**

Table 6. Results on using factors to encode morphological information for the source language when translating to English.

Starting by analyzing the results of translating to English (x-en), reported in Table 6, we can see that the factored model only resulted in an improvement for the ro-en language pair, for which we can see an increase in both BLEU and COMET scores.

Regarding the translations from English to the language with the morphology encoded in the factors (en-x), if we look at the first two columns of Table 7, we see that all the factored systems underperformed when compared to the baseline. Only for en-ro and when evaluated with BLEU, we see a slight increase, although that improvement is not reflected in the respective COMET score.

As explained in Section 3.5.1, when splitting the words into morphological lemma and respective morphological tags for the target side, once both the lemma and the factor are predicted, we must use the inflector to combine them and generate the final word. In order to evaluate if the lacking in performance was due to an underperforming inflector, we computed BLEU solely based on lemmas. To do so, we lemmatized the reference and removed all the factors from the FNMT hypothesis. In order to have something to compare this to, we lemmatized all the baseline hypothesis as well. This evaluation is also useful to understand if the FNMT model is at least choosing the correct words. These results can be seen in the third column of Table 7. Firstly all the BLEU scores are higher when compared to the inflected corpus (first column) which is expected, as mis-inflections are not taken into account in this scenario. When comparing the factored model with the baselines we see that except for en-ro once again, the factored models are still underperforming, suggesting that despite the errors injected by the inflector and the missed predicted factors, the factored model is struggling to choose the correct words when compared to the baseline.

When looking for answers for these unsatisfying results, by analyzing the hypothesis outputs we saw that the tagger and subsequently the lemmatizer were having some unexpected behaviors especially for pronouns and determiners. Given this, the FNMT models were retrained with data that only had words represented by lemma and morphological tags if they were nouns, verbs, adjectives, or adverbs, as these groups of words are the ones where inflections are most common and relevant. If we look at the two rightmost columns of Table 7, we can see that even though the factored models got closer to the baselines, they were still underperforming, with again the exception of en-ro. This suggests that limiting the prediction of lemmas and morphological tags to that restricted group of POS words stabilized the translations, even though not enough to say that using factors to encode morphology brings benefits when translating from English, with the exception of the translation to Romanian.

Finally, comparing the overall results of the translations

		All words w/ factors			Some POS w/ factors	
		BLEU	COMET	BLEU (lemmas)	BLEU	COMET
en-de	Baseline	**24.36**	**0.3605**	**27.52**	**24.36**	**0.3605**
	Factored model	21.18	0.1527	26.15	22.40	0.2420
en-lv	Baseline	**11.82**	**-0.0296**	**14.24**	**11.82**	**-0.0296**
	Factored model	10.01	-0.2727	13.70	10.78	-0.2045
en-no	Baseline	**20.98**	**-0.4464**	**22.10**	**20.98**	**-0.4464**
	Factored model	16.97	-0.6205	20.63	18.21	-0.5462
en-ro	Baseline	19.15	**0.1858**	23.40	19.15	**0.1858**
	Factored model	**19.20**	-0.1093	**25.16**	**19.26**	-0.0568

Table 7. Results on using factors to encode morphological information for the target language when translating from English. The baseline results were duplicated between the "All words w/ factors", and "Some POS w/ factors" columns, to ease the comparison between them and the results of the factored models.

in both directions of all the language pairs used, it was already expected that we got higher results when translating to English than from English, as usually, systems that translate to English tend to score higher due to the low morphological richness of this language. Furthermore, the higher scores in the factor models that translate to English and consequentially only use factors on the source side could be explained with the fact that there is no need to use the inflector, having, therefore, one fewer step in the translation pipeline and consequently one system fewer introducing possible failures. Moreover, the fact that all the models used to predict morphology related features (tagger, lemmatizer, and inflector), have in average a 10% margin for error (Table 5), could lead to the injection of noise into the corpus and harm the quality of the translations of the factored models.

We conclude that with the current setup, representing words by their morphological lemmas and respective morphological information encoded in factors only brings benefits to translate between English and Romanian.

4. Conclusion

The main goal of this thesis was to test and evaluate the usability of the factors' code in the Marian toolkit, fulfilling the first milestone of the CEF's "User-Focused Marian" agreement. We contributed to the open source Marian's codebase with the implementation of concatenation as an option to combine lemmas and factor embeddings. We analyzed the performance of factored neural machine translation models by conducting three experiments that use factors for three different use cases.

In the first experiment, we used factors to inject custom terminology into NMT at run time. We saw that our implementation to combine lemmas and factor embeddings by concatenating them outperformed in terms of correctly translated terminology the method originally implemented in Marian, which is combining them with sum. We extended the work done by Dinu *et al*. [8], by comparing these two embedding options and also by extending their research to three other language pairs, showing positive results for all of them. We also showed the importance of tying the lemma embeddings for improving the performance of the usage of factors.

The second experiment used factors to replace the subword joining markers. We compared two methods of representing this subword splits proposed by Sennrich and Haddow [5], and Wilen and Matusov [17], and concluded that the latter is the one that leads to better results. Furthermore, we showed that even though improving the translation quality for the en-de language pair, this usage of factors does not have a positive impact on the remaining tested language pairs. This experiment was also used to do a detailed analysis regarding the different factor prediction methods available in Marian, evaluating them in terms of quality of the translations generated, and speed of both training and inference.

Finally, in the third experiment, we used factors to encode morphological information, aiming to improve the translation quality of morphologically rich languages. Considerable improvements were obtained for both translation directions of the en-ro language pair. For the remaining language pairs, the results were underwhelming, in particular when translating from English to the remaining languages.

Regarding future work, given that the implementation of concatenation as a method to combine the lemma and factor embeddings in Marian was only done for the source side, the factor decoding mechanisms already available in the codebase could be adapted to also incorporate this. Furthermore, when using factors to encode morphology, instead of gathering all the morphological information into one single factor, one could try to split the different morphological groups among different factors.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. 1

[2] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. 1

[3] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. 1

[4] Jeff A. Bilmes and Katrin Kirchhoff. Factored language models and generalized parallel backoff. In *Companion Volume of the Proceedings of HLT-NAACL 2003 - Short Papers*, pages 4–6, 2003. 1

[5] Rico Sennrich and Barry Haddow. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 83–91, Berlin, Germany, August 2016. Association for Computational Linguistics. 1, 2, 4, 7, 8, 10

[6] Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. Factored neural machine translation architectures. In *Proceedings of the International Workshop on Spoken Language Translation. IWSLT'16*, Seattle, USA, 2016. 1, 2, 3, 5

[7] Rico Sennrich, Barry Haddow, and Alexandra Birch. 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, August 2016. Association for Computational Linguistics. 1, 4, 5, 6, 9

[8] Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. Training neural machine translation to apply terminology constraints. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3063–3068, Florence, Italy, July 2019. Association for Computational Linguistics. 1, 2, 4, 6, 10

[9] Miriam Exel, Bianka Buschbeck, Lauritz Brandt, and Simona Doneva. Terminology-constrained neural machine translation at SAP. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 271–280, Lisboa, Portugal, November 2020. European Association for Machine Translation. 1

[10] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia, July 2018. Association for Computational Linguistics. 1

[11] Marcin Junczys-Dowmunt, Kenneth Heafield, Hieu Hoang, Roman Grundkiewicz, and Anthony Aue. Marian: Cost-effective high-quality neural machine translation in C++. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 129–135, Melbourne, Australia, July 2018. Association for Computational Linguistics. 1

[12] Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain, April 2017. Association for Computational Linguistics. 1

[13] Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. Sockeye: A toolkit for neural machine translation, 2017. 1

[14] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July 2017. Association for Computational Linguistics. 1

[15] Colin. G. Drury and Jiao Ma. Do language barriers result in aviation maintenance errors? In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 47, pages 46–50, Denver, Colorado, October 2003. 2

[16] Colin. G. Drury, Jiao Ma, and C. V. Marin. Language error in aviation maintenance. Technical report, University of Buffalo, August 2005. 2

[17] Patrick Wilken and Evgeny Matusov. Novel applications of factored neural machine translation. 2019. 2, 4, 5, 7, 8, 10

[18] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*, 2015. 2

[19] Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain, April 2017. Association for Computational Linguistics. 3, 5, 6

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017. 4, 5

[21] OndÅ™ej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. Findings of the 2018 conference on machine translation (wmt18). In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 272–307, Belgium, Brussels, October 2018. Association for Computational Linguistics. 5

[22] Philipp Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT. 5

[23] Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. 5

[24] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. 5

[25] Ralf Steinberger, Andreas Eisele, Szymon Klocek, Spyridon Pilos, and Patrick Schlüter. DGT-TM: A freely available translation memory in 22 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 454–459, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). 5

[26] Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, Dan Tufiş, and Dániel Varga. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May 2006. European Language Resources Association (ELRA). 5

[27] Jörg Tiedemann. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). 5

[28] Roberts Rozis and Raivis Skadiņš. Tilde MODEL - multilingual open data for EU languages. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 263–265, Gothenburg, Sweden, May 2017. Association for Computational Linguistics. 5

[29] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. 5

[30] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. 5

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 5

[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 5

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. 5

[34] Chaitanya Malaviya, Shijie Wu, and Ryan Cotterell. A simple joint model for improved contextual neural lemmatization. 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 1517–1528, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 8

[35] Ben Peters and André F. T. Martins. 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, August 2019. Association for Computational Linguistics. 8

[36] Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, and *et al*. Bouma. Universal dependencies 2.0, 2017. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. 8

[37] John Sylak-Glassman. The composition and use of the universal morphological feature schema (unimorph schema). *Johns Hopkins University*, 2016. 8