Morpheus: A Neural Network for Jointly Learning Contextual Lemmatization and Morphological Tagging

Eray Yildiz
Faculty of Computer and
Informatics Engineering
Istanbul Technical University
yildiz17@itu.edu.tr

A. Cuneyd Tantug
Faculty of Computer and
Informatics Engineering
Istanbul Technical University
tantug@itu.edu.tr

Abstract

In this study, we present *Morpheus*, a joint contextual lemmatizer and morphological tagger. *Morpheus* is based on a neural sequential architecture where inputs are the characters of the surface words in a sentence and the outputs are the minimum edit operations between surface words and their lemmata as well as the morphological tags assigned to the words. The experiments on the datasets in nearly 100 languages provided by SigMorphon 2019 Shared Task 2 organizers show that the performance of *Morpheus* is comparable to the state-of-the-art system in terms of lemmatization. In morphological tagging, on the other hand, *Morpheus* significantly outperforms the SigMorphon baseline. In our experiments, we also show that the neural encoder-decoder architecture trained to predict the minimum edit operations can produce considerably better results than the architecture trained to predict the characters in lemmata directly as in previous studies. According to the SigMorphon 2019 Shared Task 2 results, *Morpheus* has placed 3^{rd} in lemmatization and reached the 9^{th} place in morphological tagging among all participant teams.

1 Introduction

Lemmatization is the process of reducing an inflected word into its dictionary form known as the lemma. Morphological tagging, on the other hand, is the process of marking up words with their morphological information and part of speech (POS) tags. Lemmatization and morphological tagging are essential tasks in natural language processing since they usually represent initial steps of subsequent tasks such as dependency parsing (Chen and Manning, 2014; McDonald and Pereira, 2006) and semantic role labeling (Haghighi et al., 2005). Morphological information of words is utilized in various tasks including statistical machine translation (Huck et al., 2017), neural machine translation (Conforti et al., 2018) and named entity recognition (Güngör et al., 2019) to improve the performance. Morphological tagging and lemmatization is crucial especially in morphologically rich languages such as Turkish and Finnish since inflected and derived words carry a substantial amount of information such as number, person, case, tense and aspect. Moreover, lexical ambiguities can occur in highly inflectional and derivational languages such as Turkish. The correct lemma and morphological tags may differ according to the context in which words appear. As shown in table 1, the Turkish word *"dolar"* may have different lemma and morphological properties according to the context it is used.

To achieve lemmatization and morphological tagging in highly inflectional languages, traditional approaches employ finite state machines which are constructed to model grammatical rules of a language (Oflazer, 1993; Karttunen et al., 1992). Building a state machine for morphological analysis is not a trivial task and requires considerable effort necessitating linguistic knowledge. Furthermore, morphological analyzers frequently produce multiple analyzes for each word which introduces morphological ambiguities. Morphological disambiguation which is the process of selecting correct analyzes of words according to the context (Yildiz et al., 2016; Shen et al., 2016) is mostly needed after morphological analysis step. Morphological disambiguation is also a difficult problem due to the fact that it requires the classification of both lemmata and the corresponding labels. Therefore, researchers have studied language-agnostic data-driven solutions for both lemmatization and morphological tagging. In most of the studies, applying machine learning or statistical methods over morphologically an-

Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 25–34
Florence, Italy. August 2, 2019

Table 1: Different lemmata and morphological properties of Turkish word "dolar" according to the context

Turkish Sentence	English Translation	Lemma of the word "dolar"	Morhoplogical Properties of the word "dolar"
atkıyı boynuna **dolar**	She/he **wraps** the scarf around his/her neck	dola *(Eng. wrap)*	Verb, Third Person Singular, Present Tense
su kovaya **dolar**	The water **fills** the bucket	dol *(Eng. fill)*	Verb, Third Person Singular, Present Tense
dolar yine yükseldi	The **dollar** increased again	dolar *(Eng. dollar)*	Noun, Nominative, Singular

notated corpora (mostly on *UniMorph* dataset[1] (Kirov et al., 2018)) have been proposed to perform joint morphological tagging and lemmatization. One of the early studies, *Morfette* (Chrupała et al., 2008) utilized a Maximum Entropy Classifier to find lemmata and morphological tags of each word in a sentence. Two separate classifiers are employed in their architecture: one for assigning morphological tags to the words and one for predicting the shortest edit script between the surface word and its lemma. Shortest edit script is the shortest sequence of instructions (insertions, deletions, and replacements) which transforms a string into another one. In this way, the system is able to apply lemmatization to out of vocabulary words by predicting the transformation which should be applied to the surface word to obtain the lemma of the word. More recent work, namely *Lemming* (Müller et al., 2015) has out-performed *Morfette* by using a Conditional Random Field classifier to classify each candidate sequences of lemmata and morphological tags jointly. The feature space of *Lemming* differs from *Morfette* as *Lemming* also uses external lexical features such as the occurrences of a candidate lemma in a dictionary. As deep neural networks gain popularity and lead state-of-the-art results in various natural language processing tasks, sequential neural networks have been successfully employed for lemmatization and morphological tagging in recent studies (Bergmanis and Goldwater, 2018; Malaviya et al., 2019; Dayanık et al., 2018; Chakrabarty et al., 2017). Promising results are obtained through standard encoder-decoder neural architectures where inputs are the character sequences of the words and outputs are the character sequences of lemmata and morphological tags (Bergmanis and Goldwater, 2018; Dayanık et al., 2018). Neural architectures which are designed to predict the edit operations between surface words and lemmata are also proposed in recent works (Chakrabarty et al., 2017). The current state of the art is held by Malaviya et al. (2019) using a neural hard attention mechanism to align the characters of surface words and lemmata. Morphological tagging and lemmatization are jointly modeled in their architecture and a dynamic programming approach is used to maximize both morphological tagging and lemmatization scores.

In SigMorphon 2019 workshop, a shared task about morphological tagging and contextual lemmatization in nearly 100 distinct languages is organized (McCarthy et al., 2019). In this study, we propose a neural network architecture, namely *Morpheus* for SigMorphon 2019 Shared Task 2. Our architecture is inspired by *MorphNet* (Dayanık et al., 2018), which has produced promising results in Turkish using an encoder-decoder neural architecture. In *MorphNet*, all characters are represented with a vector, and word vectors are generated by using long short term memories (LSTM) over character vectors. Another bidirectional LSTM is applied over word vectors to obtain context-aware representations of each word in a sentence. An LSTM based decoder inputs context-aware word representations and produces lemmata and morphological tags, respectively. Our architecture differs from *MorphNet* as we use two separate decoders for generating lemmata and morphological tags. Another difference of our architecture is that we follow the minimum edit script prediction approach considering the promising performance outputs of prior work (Chrupała et al., 2008; Müller et al., 2015; Chakrabarty et al., 2017). The lemma decoder of our network is optimized to predict the minimum edit operations between surface words and lemmata instead of predicting the character sequences of the lemmata as in *MorphNet* and *Lematus*.

Our experiments show that predicting the minimum edit operations instead of characters improves the performance significantly on *UniMorph* dataset, which is provided in SigMorphon 2019 Shared Task 2. The performance of the proposed architecture is comparable to the current state-of-the-art system (Malaviya et al., 2019), which is provided as a strong baseline by SigMorphon 2019 organizers. All of the experiments in this paper are reproducible using the codes we

[1] https://unimorph.github.io/

make publicly available[2].

2 Method

The input of our neural network based model is a sentence containing surface form words and the outputs are edit operations between surface words and their lemmata and morphological tags assigned to the words. The problem can be defined as we are searching a function whose inputs are surface words of a sentence $f([w_o, .., w_n])$ and whose output is the set of (o_i, m_i) tuples $[(o_0, m_0), .., (o_n, m_n)]$ where o_i is the set of edit operations to generate the lemma of the surface form w_i and m_i is the set of morphological tags assigned to the surface form w_i. The overall architecture of the system is illustrated in Figure 1. The system comprises 3 neural components that are running sequentially:

- The first component generates word vectors using LSTMs over the vector representations of its characters.
- The second component generates context-aware word vectors applying bidirectional LSTMs over word vectors
- Two separate LSTM decoders accept the same context-aware word vectors. The first decoder generates edit operations between surface words and lemmata while the second decoder generates morphological tags

In the final step, lemmata are generated by applying predicted edit operations to the surface words.

2.1 Generating minimum edit operations

The proposed model is designed to predict minimum edit operations to obtain the lemma from a surface word. The fundamental edit operations are *Same*, *Delete*, *Replace*, and *Insert* operations. To find minimum edit operations between surface word and its lemma, we use a dynamic programming approach[3] which is based on *Levenshtein* distance. Some sample edit operations between surface words and lemmata are given in Figure 2 for several languages. As seen in Figure 2, *Same* and *Delete* operations have only one version whereas *Replace* and *Insert* operations have multiple versions combined with the character to be replaced or inserted. So the actual number of elements in the edit operations set are determined for each language separately by processing the training data.

Generally, the length of a lemma is shorter than or equals to the length of the corresponding surface form and consequently, the number of edit operations are usually the same as the length of the surface form. However, for some languages, lemmata longer than the corresponding surface forms are observed. Since our minimum edit prediction decoder predicts an edit operation label for each character in the surface (see Section 2.4.1 for details), it fails in generating lemmata longer than the surface forms. Thus we make some modifications over the base operations generated by standard *Levenshtein* distance based algorithm. To ensure the length of the operation labels is the same as the length of the surface words, we merge consecutive *Insert* labels in the same position into one *Insert* label with multiple characters. We also combine the *Replace* labels and the following consequent *Insert* labels in the same position into one *Replace* label with multiple characters. For example, the minimum edit operations for the Russian surface-lemma pair "видна"-"видный" have four *Same* operation labels and one Replace_ый label, respectively. Note that the last character in the surface word "а" is replaced with the character "ы" and then the character й is inserted into the end. The base labels Replace_ы and Insert_й are merged into one label Replace_ый to ensure that edit operations and surface words have the same length (Figure 2).

2.2 Word representations

The first component of our network inputs character sequences of each word in a sentence and generates vector representations of the words using an LSTM network. Let w_i represents the i^{th} word with L_i characters in a sentence and w_{ij} is j^{th} character in w_i. In our model, each character w_{ij} is represented by a vector $a_{ij} \in \mathbb{R}^{d_a}$ and we calculate the vector representation of i^{th} word $\mathbf{e}_i \in \mathbb{R}^{d_e}$ by applying an LSTM over the vector representations of its constituent characters from left to right as shown in eq. (1). The last hidden state vector of the LSTM $h_{L_i} \in \mathbb{R}^{d_e}$ is considered as the vector representation of the word w_i.

$$\mathbf{e}_i = h_{L_i} = LSTM(a_{L_i}, h_{L_{i-1}}) \quad (1)$$

[2]https://github.com/erayyildiz/Morpheus/

[3]https://github.com/faircloth-lab/python-levenshtein

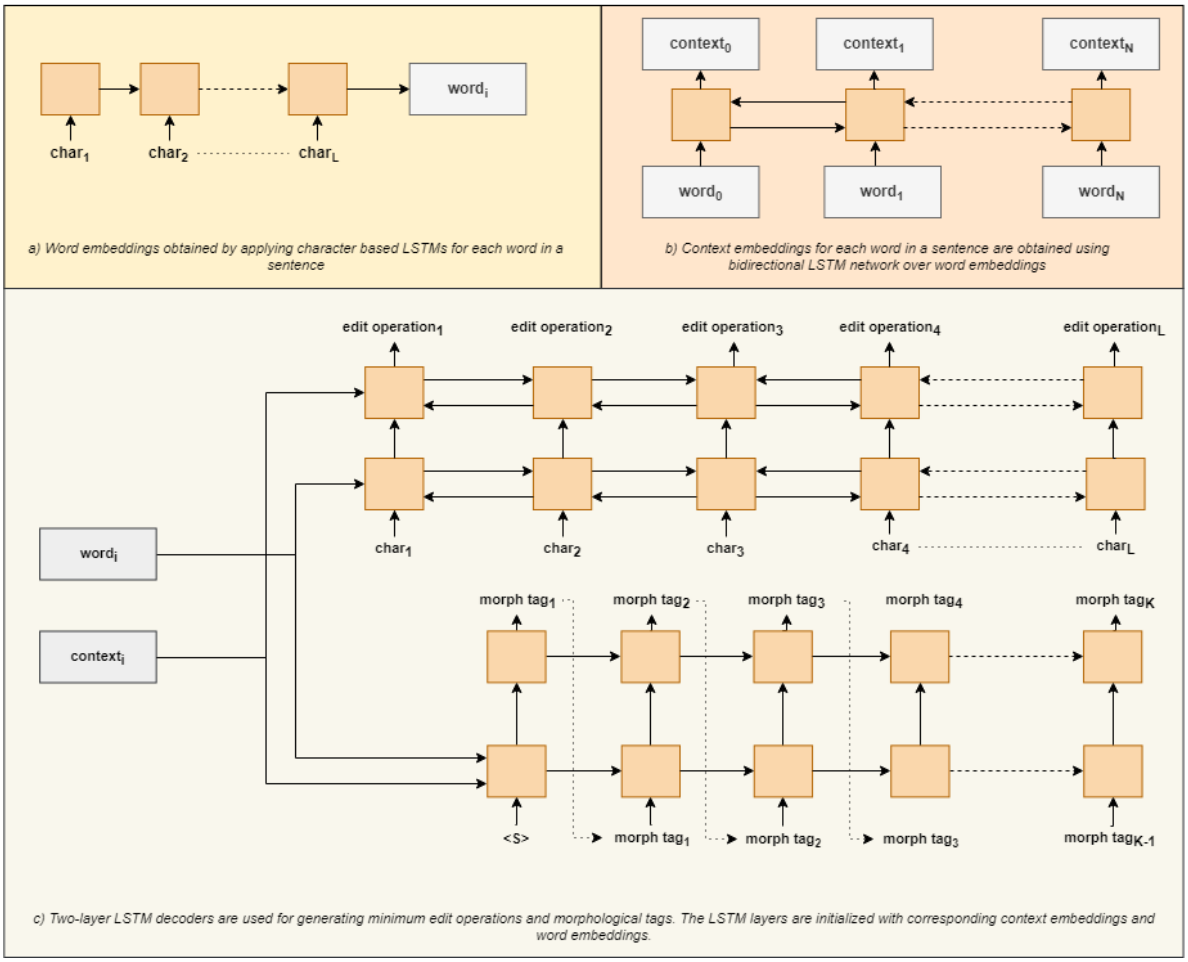

Figure 1: The illustration of the proposed neural network architecture which consists of 3 components: (a) word vector generator (b) context vector generator (c) decoders to generate minimum edit operations for lemmatization and morphological tags. (*N* indicates the number of words in the sentence. *L* indicates both the number of characters in the word and the number of edit operations between the word and the lemma. *K* represents the number of morphological tags assigned to the word)

2.3 Context-aware word representations

The context of the words have a substantial impact on morphological tagging and lemmatization in most languages (Shen et al., 2016; Malaviya et al., 2019). In order to take into account the context of the words we employ another LSTM which is bidirectional and inputs vector representations $\mathbf{e}_i$ and outputs context-aware representations $\mathbf{c}_i \in \mathbb{R}^{d_c}$ for each surface word in the context as shown in eqs. (2) to (4)

$$c_i^{\rightarrow} = LSTM(e_i, c_{i-1}^{\rightarrow}) \quad (2)$$

$$c_i^{\leftarrow} = LSTM(e_i, c_{i+1}^{\leftarrow}) \quad (3)$$

$$\mathbf{c}_i = [c_i^{\rightarrow}, c_i^{\leftarrow}] \quad (4)$$

The final output is context-aware vector representation $\mathbf{c}_i$ for each word w_i in the sentence.

2.4 Decoding Components

One of the important differences of the proposed network from previous studies (Bergmanis and Goldwater, 2018; Dayanık et al., 2018) is that it has two separate decoders for lemmatization and morphological tagging. The parameters of the decoders are not shared. However, they are both fed with the same word vectors $\mathbf{e}_i$ and context-aware word vectors $\mathbf{c}_i$ that are generated in the encoding step.

2.4.1 Minimum edit prediction decoder

The minimum edit prediction decoder component consists of a two layer bidirectional LSTM network and an embedding layer which maps each character w_{ij} in surface word to a one dimensional

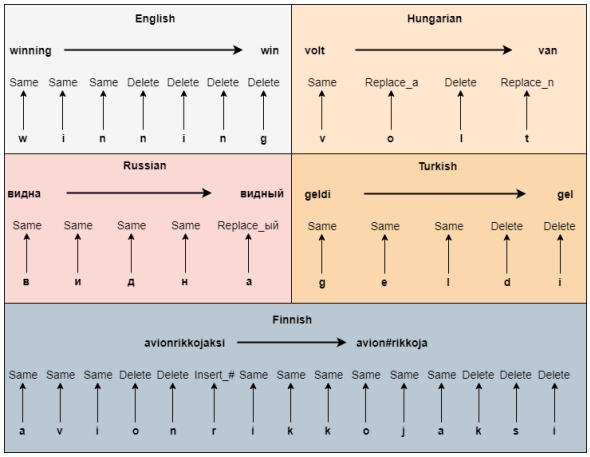

Figure 2: Minimum edit operations between surface words and their lemmata

vector $u_{ij} \in \mathbb{R}^{d_u}$. Forward LSTM network inputs previous hidden states $g_{j-1}^{\rightarrow 1}, g_{j-1}^{\rightarrow 2} \in \mathbb{R}^{d_g}$ and outputs current hidden states $g_j^{\rightarrow 1}$, $g_j^{\rightarrow 2}$ and an output vector $y_j^{\rightarrow} \in \mathbb{R}^{d_y}$. Backward LSTM network, on the other hand, applies the same operations in opposite direction and outputs $g_j^{\leftarrow 1}, g_j^{\leftarrow 2} \in \mathbb{R}^{d_g}$ and $y_j^{\leftarrow}$. *Softmax* function is then applied to the multiplication of trainable matrix $\mathbf{W}_o \in \mathbb{R}^{d_y \times |o|}$ with the concatenation vector of output vectors $y_j^{\rightarrow}$ and $y_j^{\leftarrow}$ where $|o|$ represents the number of distinct edit operations observed in the dataset. The output of *softmax* operation is the probabilities of each minimum edit operation $p(o_{ij})$ corresponding to the character w_{ij} as shown in eqs. (5) to (7).

$$y_j^{\rightarrow}, g_j^{\rightarrow 1}, g_j^{\rightarrow 2} = LSTM(u_{ij}, g_{j-1}^{\rightarrow 1}, g_{j-1}^{\rightarrow 2}) \quad (5)$$

$$y_j^{\leftarrow}, g_j^{\leftarrow 1}, g_j^{\leftarrow 2} = LSTM(u_{ij}, g_{j+1}^{\leftarrow 1}, g_{j+1}^{\leftarrow 2}) \quad (6)$$

$$p(o_{ij}) = softmax(\mathbf{W}_o \times [y_j^{\rightarrow}, y_j^{\leftarrow}]) \quad (7)$$

The first hidden states of both forward and backward LSTMs are initialized with the word vector $\mathbf{e}_i$ (see section 2.2) and a linear transformation of the context-aware word vector: $\mathbf{W}_c \times \mathbf{c}_i$ where $\mathbf{W}_c$ is a matrix in $\mathbb{R}^{d_c \times d_e}$ (see section 2.3), respectively (see eq. (11)).

$$g_0^{\rightarrow 1} = \mathbf{e}_i \quad (8)$$

$$g_0^{\rightarrow 2} = \mathbf{W}_c \times \mathbf{c}_i \quad (9)$$

$$g_{L_i}^{\leftarrow 1} = \mathbf{e}_i \quad (10)$$

$$g_{L_i}^{\leftarrow 1} = \mathbf{W}_c \times \mathbf{c}_i \quad (11)$$

2.4.2 Morphological tagging decoder

The morphological tagging decoder component is another LSTM decoder which is able to generate morphological tags without length restriction. Each word w_i has K_i morphological tags and each morphological tag m_{il} is represented by a one dimensional vector $v_{il} \in \mathbb{R}^{d_v}$. A two layer LSTM network which is unidirectional is initialized same as in minimum edit prediction component. An LSTM cell inputs the vector representation $v_{i(l-1)}$ of previously predicted tag m'_{il} and previous hidden states $q_{l-1}^1, q_{l-1}^2 \in \mathbb{R}^{d_q}$ in each step. The outputs of the LSTM cell are current hidden states q_l^1, q_l^2 and an output vector $z_{ij} \in \mathbb{R}^{d_z}$. *Softmax* function is then applied to multiplication of the output vector z_{ij} and trainable matrix $\mathbf{W}_m \in \mathbb{R}^{d_z \times |m|}$ where m equals to the number of distinct morphological tags in the dataset. The first input to the decoder is the vector representation of a special start symbol $v_{start} \in \mathbb{R}^{d_v}$. In this way the probabilities of each morphological tags $p(m_{il})$ in given position i, l are calculated as shown in eqs. (5), (12) and (13).

$$z_1, q_1^1, q_1^2 = LSTM(v_{start}, \mathbf{e}_i, \mathbf{W}_c \times \mathbf{c}_i) \quad (12)$$

$$z_l, q_l^1, q_l^2 = LSTM(v_{i(l-1)}, q_{l-1}^1, q_{l-1}^2) \quad (13)$$

$$p(m_{il}) = softmax(\mathbf{W}_m \times z_l) \quad (14)$$

2.5 Character prediction decoder

The character prediction decoder component which sequentially predicts the characters occur in lemmata is not employed in the proposed architecture. However, we build an alternative model in which the character prediction decoder component is used instead of a minimum edit prediction decoder. In this way, we aim to evaluate the impact of predicting minimum edit operations instead of characters in lemmata. The character prediction decoder used in the experiments has the same architecture and parameter set with the morphological tagging decoder.

2.6 Training objective

All the parameters in whole architecture including all LSTM parameters and the trainable matrices $\mathbf{W}_c$, $\mathbf{W}_o$, $\mathbf{W}_m$ are optimized jointly in training

Method	Lemmatization Accuracy (%)	Morphological Tagging F1 Score (%)
Turku NLP (Kanerva et al., 2018)	92.18	86.7
UPPSALA Uni. (Moor, 2018)	58.5	88.32
SigMorphon 2019 Baseline (Malaviya et al., 2019)	93.95	68.72
Morpheus (Character Prediction)	88.03	88.94
Morpheus (Edit Operation Prediction)	**94.15**	**90.52**

Table 2: Average lemmatization and morphological tagging performances of the systems on *UniMorph* dataset

phase by minimizing the sum of two cross entropy losses as follow:

$$\mathcal{L}_{lemma} = -\frac{1}{N}\sum_{i}^{N}\frac{1}{L_i}\sum_{j}^{L_i}\log p(o_{ij}) \quad (15)$$

$$\mathcal{L}_{morph} = -\frac{1}{N}\sum_{i}^{N}\frac{1}{K_i}\sum_{j}^{K_i}\log p(m_{ij}) \quad (16)$$

$$\mathcal{L}_{total} = \mathcal{L}_{lemma} + \mathcal{L}_{morph} \quad (17)$$

The loss for lemmatization is calculated by taking cross entropy over predicted minimum edit operations $p(o_{ij})$ as in eq. (15) where N stands for the number of words in the sentence and L_i stands for the number of characters in the word w_i. The loss for morphological tagging is cross entropy over predicted tag probabilities $p(m_{il})$ as in eq. (16) where K_i stands for the number of morphological tags assigned to the word w_i. The total loss to minimize is the sum of the lemmatization loss and morphological tagging loss (see eq. (17)).

3 Experiments

In our experiments, we train and evaluate the proposed architecture on *UniMorph* dataset collection (Kirov et al., 2018) for each language. Same architecture with the same hyper-parameters is used for all the languages. To investigate the impact of the minimum edit prediction component on the performance, we also train the network in which the character prediction decoder component is used instead of a minimum edit prediction component. The results of the experiments are provided in section 3.3 and table 2

3.1 Dataset

UniMorph dataset collection, which includes a various number of sentences consist of surface words with annotated lemmata and morphological tags in 97 different languages are provided in SigMorphon 2019 shared task 2. The dataset for each language is split into train and validation sets, and the size of the dataset differs in each language. In our experiments, we train our architecture on train sets and evaluate the performances on validation sets. The experimental results presented in section 3.3 are obtained on the validation sets. Note that final results which are presented in SigMorphon paper (McCarthy et al., 2019) are calculated over the test sets which were not available to the systems before the final submission stage.

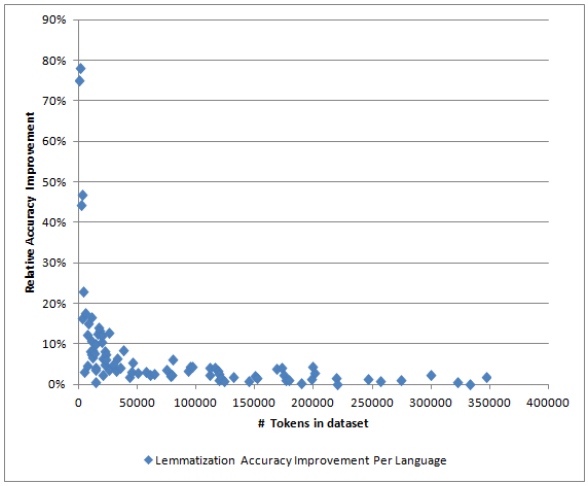

Figure 3: Relative lemmatization accuracy improvement vs dataset size per language

3.2 Experimental Setup

The same settings are used in the training of the architecture for each language. The input character embedding length d_a is set to 128 while the length of the word vectors d_e is set to 1024 and the length of the context-aware word vectors d_c is set to 2048. The length of character vectors in the minimum edit prediction component d_u and the length of the morphological tag vectors d_v are set to 256 while the hidden unit sizes in decoder LSTMs d_g and d_q are set to 1024. We use *Adam* optimization algorithm (Kingma and Ba, 2014) with learning rate $3e-4$ for minimizing the loss

Language	Dataset Size	Lemmatization Accuracy (%)		Morph. Tagging F1 (%)	
		Character Pred.	Edit Pred.	Character Pred.	Edit Pred.
North-Sami-Giella	29K	87.53	**91.90**	88.89	**92.83**
French-GSD	360K	97.06	**98.47**	97.58	**97.99**
Japanese-Modern	14K	85.39	**93.88**	**93.06**	92.44
Swedish-PUD	18K	82.79	**93.05**	89.23	**92.09**
Arabic-PADT	256K	94.39	**95.18**	95.01	**95.40**
Basque-BDT	119K	95.42	**96.49**	93.06	**94.47**
Urdu-UDTB	123K	95.20	**96.02**	90.79	**91.20**
Irish-IDT	21K	85.07	**89.23**	**80.60**	71.52
Bambara-CRB	14K	88.24	**88.85**	93.47	**93.56**
Dutch-Alpino	200K	94.97	**97.81**	95.63	**96.45**
Czech-FicTree	175K	97.39	**98.49**	94.15	**96.39**
Danish-DDT	94K	93.16	**97.26**	94.17	**95.62**
Latin-ITTB	332K	98.65	**98.75**	96.84	**97.34**
French-Sequoia	64K	95.54	**98.17**	95.96	**96.82**
Italian-PoSTWITA	115K	92.71	**96.61**	94.43	**95.62**
Polish-SZ	93K	93.59	**96.86**	90.23	**93.26**
Czech-CLTT	32K	92.11	**98.03**	89.03	**93.82**
Cantonese-HK	7K	90.05	**94.17**	85.41	**86.14**
Galician-TreeGal	23K	89.68	**95.19**	89.78	**90.72**
Slovenian-SSJ	131K	95.25	**96.94**	93.47	**95.79**
French-ParTUT	25K	92.67	**96.10**	93.09	**94.55**
Lithuanian-HSE	5K	70.60	**83.05**	43.37	**70.70**
French-Spoken	35K	94.47	**98.48**	95.46	**96.66**
Russian-Taiga	22K	83.59	**90.57**	76.62	**83.80**
Latvian-LVTB	150K	94.29	**96.22**	93.51	**95.87**
Czech-PDT	1515K	84.86	**98.41**	87.65	**95.27**
Japanese-GSD	168K	95.21	**98.91**	95.35	**95.61**
Indonesian-GSD	111K	97.06	**99.49**	92.69	**93.11**
Gothic-PROIEL	62K	**96.60**	96.58	93.04	**95.12**
Latin-PROIEL	219K	96.31	**96.37**	93.75	**95.05**
Czech-PUD	19K	83.55	**93.57**	81.30	**86.70**
Dutch-LassySmall	96K	93.44	**97.58**	94.51	**95.47**
Romanian-RRT	198K	96.54	**97.88**	96.81	**97.44**
Korean-Kaist	346K	93.31	**95.07**	95.70	**95.46**
Amharic-ATT	11K	93.80	**100.00**	91.02	**91.39**
English-GUM	79K	95.58	**97.85**	93.92	**95.48**
Estonian-EDT	421K	93.10	**96.27**	95.64	**96.70**
Chinese-GSD	111K	95.22	**99.15**	89.25	**90.78**
Korean-GSD	80K	87.55	**92.89**	93.43	**94.16**
Marathi-UFAL	4K	74.59	**76.94**	68.26	**69.26**
Akkadian	2K	42.22	**60.89**	**78.13**	66.52
Faroese-OFT	13K	83.56	**89.97**	88.08	**89.49**
English-EWT	246K	96.78	**98.11**	95.61	**95.95**
Sanskrit-UFAL	3K	53.61	**65.98**	52.59	**55.36**
Turkish-IMST	60K	94.13	**96.43**	91.67	**93.72**
English-PUD	20K	89.40	**95.22**	88.88	**89.89**
Korean-PUD	18K	87.19	**98.86**	91.42	**92.75**
Finnish-PUD	16K	77.72	**87.55**	85.49	**92.05**
Russian-SynTag	1036K	95.31	**97.76**	94.99	**95.13**
Croatian-SET	179K	94.91	**96.01**	94.31	**95.47**
Tagalog-TRG	406	48.00	**84.00**	**74.23**	71.74
Slovenian-SST	31K	91.83	**94.97**	85.34	**89.23**
Finnish-FTB	172K	90.70	**94.46**	94.13	**95.85**
Polish-LFG	174K	93.85	**96.09**	92.93	**95.35**
Portuguese-Bosque	218K	96.43	**97.86**	96.07	**96.59**
Coptic-Scriptorium	20K	93.47	**95.68**	95.17	**94.76**
Chinese-CFL	7K	82.55	**92.66**	81.51	**83.76**
Spanish-AnCora	497K	98.32	**98.92**	98.29	**98.46**
Greek-GDT	57K	93.73	**96.65**	94.71	**96.12**
Serbian-SET	78K	94.82	**97.06**	94.36	**96.06**
Naija-NSC	14K	95.80	**99.84**	91.15	**92.02**
Vietnamese-VTB	42K	98.17	**99.95**	89.45	**89.71**
Yoruba-YTB	2K	83.60	**97.20**	**80.49**	70.67
Italian-PUD	22K	89.51	**96.11**	92.63	**94.22**
Finnish-TDT	198K	91.37	**95.38**	95.67	**96.76**
English-ParTUT	44K	94.87	**97.85**	92.32	**93.46**
Upper-Sorbian-U.	11K	77.79	**90.74**	69.47	**77.46**
Norwegian-Ny.	14K	93.89	**97.42**	90.45	**92.20**
Galician-CTG	121K	97.18	**98.69**	97.29	**97.30**
Old-Church-Slv.	66K	96.48	**95.66**	93.33	**94.91**
Russian-GSD	92K	**92.90**	91.51	91.98	**93.91**
Kurmanji-MG	10K	85.66	**92.69**	**85.99**	85.22
Norwegian-Bk.	299K	96.65	**98.94**	96.75	**97.41**
Italian-ISDT	273K	96.90	**97.90**	97.34	**97.89**
Komi-Zyrian-IKDP	1K	38.55	**68.67**	**45.50**	36.89
Hebrew-HTB	144K	96.49	**97.35**	95.35	**95.70**
Tamil-TTB	10K	86.77	**96.10**	83.07	**88.50**
Buryat-BDT	10K	83.33	**89.61**	78.24	**82.30**
Breton-KEB	12K	85.61	**92.81**	88.65	**90.12**
Latin-Perseus	29K	**87.30**	86.26	79.21	**82.88**
Romanian-Nonstd	189K	96.10	**96.37**	95.52	**96.36**
Italian-ParTUT	50K	94.65	**97.44**	94.83	**96.30**
Catalan-AnCora	481K	98.17	**98.92**	98.42	**98.65**
Arabic-PUD	22K	**81.31**	80.90	87.23	**88.00**
Komi-Zyrian-L.	2K	52.75	**77.47**	55.79	**57.02**
Japanese-PUD	25K	86.30	**97.32**	93.46	**94.02**
Slovak-SNK	119K	94.52	**96.95**	91.88	**94.55**
Ukrainian-IU	118K	93.63	**96.80**	91.24	**93.68**
Turkish-PUD	17K	78.20	**89.19**	86.71	**91.28**
Bulgarian-BTB	152K	95.98	**97.58**	97.16	**97.83**
Russian-PUD	19K	83.78	**92.54**	81.73	**87.79**
Belarusian-HSE	8K	78.06	**89.87**	69.39	**71.95**
Hindi-HDTB	322K	98.15	**98.82**	96.12	**96.60**
Czech-CAC	474K	98.01	**98.86**	96.52	**97.54**
Hungarian-Szeged	38K	87.89	**95.26**	89.63	**91.65**
Swedish-LinES	74K	93.52	**96.82**	93.25	**94.88**
Afrikaans-Af.B.	45K	93.75	**98.74**	95.08	**95.96**
English-LinES	77K	96.19	**98.27**	94.57	**95.43**

Table 3: Lemmatization and Morphological Tagging performances of minimum edit prediction model and character prediction model on development sets

function. *Dropout* rate 0.4 is applied to all connections during model training for regularization. All the weights are initialized with *Xavier* initialization method (Glorot and Bengio, 2010). We use an early stop mechanism which stops the training after four consecutive epochs without improvement on validation set.

3.3 Results

Table 2 presents the lemmatization and morphological performances of the proposed method on UniMorph dataset collection. The lemmatization accuracy on a dataset is the proportion of the number of correctly found lemmata over the total lemmata count. The lemmatization accuracy given in table 2 is the average of the accuracies obtained over the validation sets of all languages. The performance of morphological tagging is measured by the F1 score calculated over the predicted and actual individual morphological tags. In addition to the performance of the proposed architecture with minimum edit prediction decoder, the performance of the architecture with character prediction decoder is also given. The performances of SigMorphon 2019 neural baseline, *Turku NLP* system (Kanerva et al., 2018) which is the best lemmatization performer in CONLL 2018 Shared Task (Zeman and Hajič, 2018) and *UPPSALA Uni* system which is the best morphological tagging performer in CONLL 2018 are also given. Although the dataset provided in CONLL 2018 share the same basis with the dataset provided in SigMorphon, important differences exist between them. Hence the performances of *Turku NLP* and *UPPSALA Uni* are not directly comparable to our systems and SigMorphon baselines. However, we present the performances of those systems averaged on the same languages in SigMorphon dataset to provide an idea of how much improvement is achieved over a year.

According to the results, the proposed architecture, *Morpheus* performs slightly better than the SigMorphon neural baseline in terms of the average lemmatization accuracy. Similarly, for the morphological tagging task, *Morpheus* with a minimum edit prediction decoder significantly outperforms the baseline and *Morpheus* with character prediction decoder The experiments show that the performance is improved considerably when the minimum edit prediction decoder is used instead of the character prediction decoder. An im-

pressive result is that the performance of morphological tagging is also enhanced by employing the minimum edit prediction decoder.

Table 3 shows the lemmatization and morphological performances of both character prediction and minimum edit prediction models for each language. The performance of the minimum edit prediction model is better than the character prediction model in almost all languages. Figure 3 shows that there is a correlation between the size of the training data and the improvement on the performance when the minimum edit prediction decoder is employed. For instance, the relative lemmatization improvement is extreme in languages with relatively small dataset such as Tagalog-TRG (400 tokens/0.75 relative improvements), Komi-Zyrian (1.1K tokens/0.78 relative improvement) and Akkadian (1.7 tokens/0.44 relative improvement). On the other hand, in languages with large size dataset such as Spanish-AnCora (496K tokens), Catalan-AnCora (480K tokens) and French-GSD (359K tokens), the improvement is relatively low (0.006, 0.007 and 0.01, respectively). Although improvement magnitude is highly correlated with the training dataset size, there must be other factors specific to the properties of the language. For instance, the dataset size of the language Marathi-UFAL is small (4.1K tokens). However, the improvement degree is also small (0.03 relative improvement).

Language	**Surface word**	**Edit pred. based model output**	**Char. pred. based model output**
English	Thurlow Cosmic sorrows Vietnam	Thurlow Cosmic sorrow Vietnam	Throughlough Comsic sorw Vietman
Turkish	YPK kokainle Islık	Ypk kokain ıslık	YYk koki sılık

Table 4: Some examples of the errors made by character prediction model and corrected by the edit prediction based model

To investigate in which cases the edit prediction model performs better, we explore the outputs of the models for English and Turkish languages. A significant portion of the errors in the character prediction model is observed in unseen words and proper nouns. Some of the errors made by the character prediction based model and corrected by the edit prediction based model are shown in Table 4. A possible reason is that the lemmatization of a singular nominative noun which is rarely seen in the training data is easier to edit prediction model since all of the edit operations are *Same* and the model should only produce a sequence of *Same* labels. Character prediction based model, on the other hand, have to learn to reproduce the word from scratch. Additionally, we observe a significant amount of samples in which the edit prediction model produced morphological tags and lemmata more appropriate to the context than the outputs of the character prediction model. As a result, further research is needed to understand in which cases the edit prediction decoder helps to better learning of morphological properties of a language.

4 Conclusion

In this study, we propose a neural architecture, namely *Morpheus*, which is based on sequential neural encoder-decoders. The input words are encoded in context-aware vector representations using two-level LSTM network and the decoders initialized with context-aware word vectors generates both morphological tags assigned to the words and minimum edit operations between surface words and their lemmata. We perform experiments to evaluate the performance of *Morpheus* on *UniMorph* dataset collection (Kirov et al., 2018), which comprised nearly 100 language datasets. The experiments show that the lemmatization performance of the *Morpheus* is comparable to the SigMorphon neural baseline system (Malaviya et al., 2019), which has obtained current state-of-the-art results on *UniMorph* dataset collection. Regarding morphological tagging performance, *Morpheus* outperforms SigMorphon morphological tagger baseline significantly (0.3 relative improvement). In lemmatization, *Morpheus* has placed 3^{rd} in the SigMorphon 2019 Shared Task 2, and it has reached the 9^{th} place in morphological tagging. In our experiments, we also show that predicting the minimum edit operations between surface words and their lemmata instead of directly predicting the characters improves the performances of the system significantly especially when the dataset is small.

Acknowledgments

We would like to thank the SigMorphon 2019 organizers for the great effort and the reviewers for the insightful comments.

References

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with lematus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1391–1400.

Abhisek Chakrabarty, Onkar Arun Pandit, and Utpal Garain. 2017. Context sensitive lemmatization using two successive bidirectional gated recurrent networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1481–1491.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Grzegorz Chrupała, Georgiana Dinu, and Josef Van Genabith. 2008. Learning morphology with morfette.

Costanza Conforti, Matthias Huck, and Alexander Fraser. 2018. Neural morphological tagging of lemma sequences for machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, volume 1, pages 39–53.

Erenay Dayanık, Ekin Akyürek, and Deniz Yuret. 2018. Morphnet: A sequence-to-sequence model that combines morphological analysis and disambiguation. *arXiv preprint arXiv:1805.07946*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Onur Güngör, Tunga Güngör, and Suzan Üsküdarli. 2019. The effect of morphology in named entity recognition with sequence tagging. *Natural Language Engineering*, 25(1):147–169.

Aria Haghighi, Kristina Toutanova, and Christopher D Manning. 2005. A joint model for semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 173–176. Association for Computational Linguistics.

Matthias Huck, Aleš Tamchyna, Ondřej Bojar, and Alexander Fraser. 2017. Producing unseen morphological variants in statistical machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 369–375.

Jenna Kanerva, Filip Ginter, Niko Miekka, Akseli Leino, and Tapio Salakoski. 2018. Turku neural parser pipeline: An end-to-end system for the conll 2018 shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 133–142.

Lauri Karttunen, Ronald M Kaplan, and Annie Zaenen. 1992. Two-level morphology with composition. In *COLING 1992 Volume 1: The 15th International Conference on Computational Linguistics*, volume 1.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sebastian J. Mielke, Arya McCarthy, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. UniMorph 2.0: Universal Morphology. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Chaitanya Malaviya, Shijie Wu, and Ryan Cotterell. 2019. A simple joint model for improved contextual neural lemmatization. *arXiv preprint arXiv:1904.02306*.

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

Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*, pages 81–88.

Christophe Moor. 2018. *Multilingual Dependency Parsing from Raw Text to Universal Dependencies: The CLCL entry*. Ph.D. thesis, University of Geneva.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274.

Kemal Oflazer. 1993. Two-level description of turkish morphology. In *Proceedings of the Sixth Conference on European Chapter of the Association for Computational Linguistics*, EACL '93, pages 472–472, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell, and Chris Dyer. 2016. The role of context in neural morphological disambiguation. In *COLING*, pages 181–191.

Eray Yildiz, Caglar Tirkaz, Bahadir Sahin, Mustafa Tolga Eren, and Ozan Sonmez. 2016. A morphology-aware network for morphological disambiguation.

Daniel Zeman and Jan Hajič, editors. 2018. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Brussels, Belgium.