Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 192 (2021) 49–58

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Deep Learning Approach for Automatic Romanian Lemmatization

Maria Nuţu[a,b]

[a]Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, RO-400084
[b]Communications Department, Technical University of Cluj-Napoca, RO-400114

Abstract

This paper proposes a deep learning sequence-to-sequence approach to improve the task of automatic Romanian lemmatization. The study compares 24 systems using different combinations of recurrent, convolutional and attention layers, while the text input consists of word-lemma pairs, both at word and trigram level. As Romanian is a low resourced language in the field of text processing, the aim of this study is to use as little input information as possible. Thus, to increase the lemmatization accuracy, additional lexical features (such as context and POS tags) have been provided only gradually. For the trigrams case, two scenarios were proposed: to predict the lemma for every word from the sequence or to predict the lemma only for the word in the middle.

The lemmatizers have been analyzed on two Romanian datasets: the Romanian Explicative Dictionary (DEX) and the belletristic subset of the CoRoLa corpus. For the DEX dataset, the best results were obtained with the LSTM-based systems at both word (99.32%) and character level (99.43%). For the CoRoLa subset, the CNN-based architecture outperforms at trigram (95.86%) and word level (99.09%) while the LSTM-stacked system obtained the highest accuracy at character level (98.78%).

Keywords: Lemmatization, Part-of-Speech tagging, LSTM, CNN, deep neural networks, sequence-to-sequence, Romanian

1. Introduction

Recent works in the deep learning field focus on using as few input features as possible, offering an end-to-end strategy fed only with (labeled) data needed for the task. For instance, in the text-to-speech scenario [23, 27], to create synthetic voices, only pairs of written text and corresponding audio are necessary and it is the system's responsibility to learn a mapping between lexical and acoustic features. Although it seems that no additional textual or lexical data is needed, the text processing tasks, such as phonetic transcription [28], lemmatization [5, 30] or POS tagging [20] can improve the quality of the synthesised speech.

The study addresses lemmatization, as this intends to be a preliminary text processing step in the task of creating synthetic voices with emotional expressivity. Romanian is a low resourced language considering the lack of emotional

E-mail address: maria.nutu@cs.ubbcluj.ro (Maria Nuţu).

10.1016/j.procs.2021.08.006

labeled corpora, thus it is a challenge to develop a method to automatically label the text based on the emotional content level. A range of studies addresses the connection between the word's lemma and emotions within a text. In [22] the lemma is used to measure the level of emotion in political discourse, but the method can be extended to other domains. The authors of [9] and [18] propose lemma-based approaches to analyze the texts for preventing a suicide behaviour. In [2] lemma is used to create an emotion lexicon. Apart from detection emotion in texts, lemma is efficient to reduce the complexity of the vocabulary in fields like topic identification [11], text summarisation [24], word search [15], machine translation [13] or keyword spotting [15].

In the current experiments three types of Deep Neural Networks architectures have been compared. The Long Short-Term Memory (LSTM) cells and Convolutional Neural Networks (CNN) with Attention layer have been chosen to train 24 systems with three types of input-output data: simple pairs composed of word and corresponding lemma, tuples of words enriched with morphological information and the corresponding lemma and, finally, trigrams (sequence of three words, with or without morphological description) and the lemmas. In this last scenario, lemmas were predicted either for the entire sequence of three words or only for the word in the middle of the trigram. The analyzed data is part of two Romanian corpora: the Romanian Explicative Dictionary (DEX)[1] and CoRoLa corpus [16].

The paper is structured as follows: Section 2 summarizes the recent work within the Romanian lemmatization problem. A concise theoretical background of lemmatization is described in Section 3, while the method used in the current experiments is detailed in Section 4. The experimental setup is analysed in Section 5. Results are discussed in Section 6, while conclusions are drawn in Section 7.

2. Previous work

In recent years a large number of research papers have analysed the lemmatization problem using deep learning strategies. Table 1 and the following related work discussion summarizes the different studies in the Romanian lemmatization field. However, due to the different datasets, data versioning and experimental details, the results are not directly comparable.

Kanerva et al. [14] use a two BiLSTM layered encoder, enriched with learned character and POS tag embeddings, and a decoder composed of two unidirectional LSTM layers with an input feeding attention on top of the encoder's output. The system was trained for 52 different languages. The **accuracy** of the Romanian lemmatizer was **98.25%**.

Qi et al. [21] present *Stanza*, a Python multilingual NLP tool, processing 66 languages. To predict the lemma, *Stanza* uses a pair of a dictionary-based and a sequence-to-sequence lemmatizers with LSTM cells and attention mechanism. For Romanian language, an accuracy of **97.95%** was obtained for the Romanian-RRT dataset.

Chrupala et al. [7] describe *Morfette*, a modular system for morphological tagging and lemmatization, based on searching algorithms, the shortest sequence of instructions (shortest edit script - SES [17]) and Maximum Entropy classifiers. The two modules can be used together to improve the input information (one's input uses the other's output) or separately. The lemmatizer's input consists of word-lemma pairs enriched with lexical features (prefixes and suffixes of a certain length, predicted POS, spelling pattern). *Morfette* is analysed on three morphologically rich languages: Polish, Spanish and Romanian. For the Romanian language the lemmatizer was trained on the MULTEXT-EAST[2] dataset obtaining an **accuracy** of **97.78%**.

In [29] Yildiz et al. present *Morpheus*, a lemmatizer and morphological tagger. The tools follows the encoder-decoder architecture, using LSTM cells and dedicated decoder for each task. The lemmatizer used an extra bidirectional LSTM to encapsulate the word's context. Unlike most lemmatizers which directly predict the characters of the word's lemma, *Morpheus* also outputs the minimum edit operations between the word and its lemma. For Romanian language, *Morpheus* achieves an **accuracy** of **97.88%** for edit operations and **96.54%** when predicting characters. The Romanian-RRT dataset was used.

[1] https://dexonline.ro/
[2] http://nl.ijs.si/ME/

The NLP-Cube[3] framework described in [4] performs sentence splitting, tokenization, lemmatization, tagging and parsing for 82 languages. For the lemmatization task, the system is composed of a bidirectional LSTM encoder and a simple LSTM decoder leading to a **94.79% accuracy** when applied to the Romanian language.

Chakrabarty et al. [6] use two successive BiLSTM structures to perform a context sensitive language independent lemmatization. The first BiLSTM network extracts the character level dependencies, while the second network learns contextual information for the given word. For the Romanian language an **accuracy** of **94.32%** was obtained.

In [3] Boroş implements a framework based on a perceptron-like algorithm with a margin-dependent learning rate (MIRA [8]) and solves NLP tasks such as syllabification, lemmatization, phonetic transcription and lexical stress prediction applied on Romanian language. For lemmatization the overall **accuracy** was equal to **94%** (initially, the accuracy was computed separated for every POS class).

In [25] Straka et al. compare the contribution of three conceptualized embeddings (BERT[10], Flair[1] and ElMo[19]) within a LSTM based system enriched with word embeddings (WE) and character-level word embeddings (CWE). The analyzed tasks were POS tagging, lemmatization and dependency parsing for a number of 54 languages. In the particular task of lemmatization in Romanian language, the Romanian-RRT[4] (ro-rrt) dataset was chosen. The highest **F1-score** of **98.59%** was achieved when both BERT and Flair embeddings were used.

Table 1: Literature results for Romanian Lemmatization.

Reference	Dataset	Context	Architecture	Accuracy
Kanerva et al.[14]	ro-rrt	POS taggs	BiLSTM + LSTM	98.25
Qi et al. [21]	ro-rrt	LSTM + dict.	LSTM + Attention	97.95
Chrupala et al. [7]	MULTEXT-EAST	POS + lexical feats	search alg. + classifiers	97.78
Yildiz et al. [29]	ro-rrt	BiLSTM+ vector repres.	LSTM + BiLSTM	96.54
Boroş [4]	ro-rrt	POS+WE	BiLSTM + LSTM	94.79
Chakrabarty et al. [6]	ro-rrt	semantic embedd.	BiLSTM	94.32
Boroş [3]	DEX	POS	Perceptron	94
Straka et al. [25]	ro-rrt	BERT+WE+CWE	LSTM	98.59 (F1-score)

3. Lemmatization - theoretical background

Lemmatization is the process of determining the lemma (the word's dictionary form) of a certain word. In the linguistic fields, through lemmatization, all flexional forms of a word are grouped together to be analysed as a single entity. The lemmatization is language dependent and adheres to certain rules. In Romanian, the lemma of a noun is often the masculine singular nominative, while a verb's lemma is the infinitive form.

Noun *copilandre (plural,feminine)→ copilandru (singular, masculine)* = youth
Verb *merg = (I) go, mergeam = (I) went, mersesem = (I) had gone* → merg = to go

In contrast to stemming, which returns the part of the word that never changes even when different forms of the word are used (the stem), lemmatization depends on the word's meaning or context and on the morphology of the word (Table 2). Although the stemming algorithms are faster and easier to be applied in word searching applications, they have lower accuracy in the case of homonyms (words spelled the same but with different meaning). For instance, the word *ouă* (*En: eggs)* has the same stem *ou* (*En: egg*), in both of the examples (illustrated in Table 3), regardless of the part of speech, while the lemma differs based on the word's meaning and part of speech.

Ambiguous words. Based on semantical and morphological context, the lemmatization methods can be grouped in two major categories: (I) *context-aware methods* - the system knows information regarding the context in which the

[3] https://github.com/adobe/NLP-Cube
[4] https://universaldependencies.org/treebanks/ro_rrt/indthe corpora being shuffled before or after the splittingex.html

Table 2: The following family of words preserves the stem, while the lemma differs.

Word	Lemma	Stem
copilașul = infant	copilaș	
copilandru = youth	copilandru	
copilărie = childhood	copilărie	copil = child
copilări = to childhood	copilări	
copiliță = little girl	copiliță	
copilărește = childishly	copilăresc	

Table 3: The word ***ouă*** preserves the stem while the lemma depends on the context and on the POS.

Example	POS	Lemma	Stem
Am cumpărat patru **ouă**. *En: I bought four eggs.*	Noun	ouă → ou	ouă → ou (*En: egg*)
Găinile **ouă**. (*En: The hens lay eggs*	Verb	ouă → (a) oua	ouă → ou (*En. to lay eggs*)

word appears (sentence context - for meaning, POS tag- for morphological information, etc.) and (II) *individual word-based methods* - the system predicts the lemma knowing only the given word, without any additional information. The advantage of the first approach is the higher accuracy rate when predicting lemma for ambiguous words, as the system can benefit from the contextual information provided. The second strategy can be improved by listing all the possible lemmas of the given words. In this way, if the predicted lemma belongs to the word's list of lemmas, then it will be considered correct, as the system's total unawareness of the context was taken into consideration.

Being a rich inflectional and morphological language, the Romanian ambiguous words form a consistent category which is challenging for the lemmatization task. The ambiguity can appear either between different POS classes (as an adverb, *poate = maybe* has the associated lemma "poate", while, as verb *poate = (he) can* has "putea" as lemma) or within the same POS class (the plural noun *torturi* means both *cakes* and *tortures*, depending on the pronunciation; thus it has two lemmas - *tort* and *tortură* respectively).

4. Method Overview and architectures

Sequence-to-sequence learning. The task of lemmatization assumes working with variable length input and output sequences, as the given word might be longer than its lemma. The sequence-to-sequence (seq2seq) [26] architecture is frequently applied in these situations. It is formed of two neural networks: an *encoder* and a *decoder*. The encoder is responsible for discerning the input and representing it in a lower dimensional space. The output of the encoder will then be used to condition the decoding network's prediction. Figure 1 presents a seq2seq model for the word "are"(*En. has*) as input and "avea" (*En. to have*) as output. The special tags <SS> and <SE> mark the start and the end of the sequence. The most prevalent architectures behind the encoders/decoders are the recurrent and convolutional neural networks.

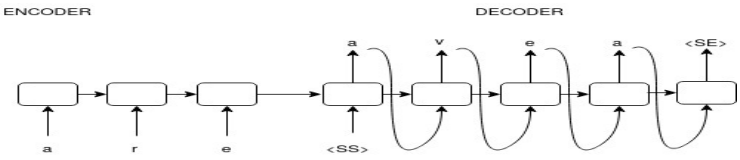
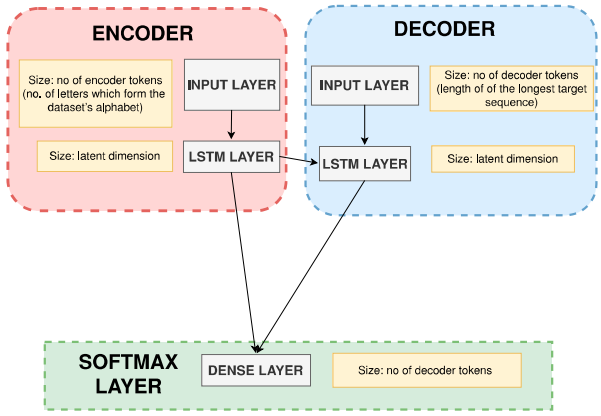
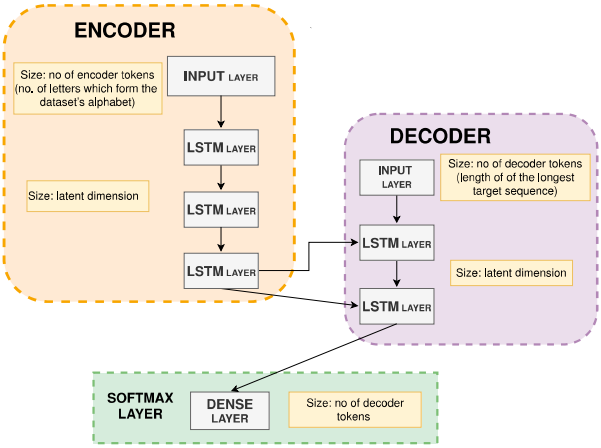


Fig. 1: Sequence-to-sequence flow

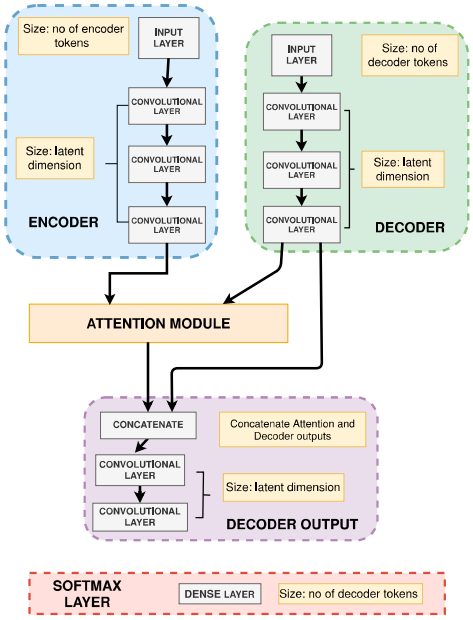
Recurrent Neural Networks. As a particular type of artificial neural network, the Recurrent Neural Networks (RNN) use as input not only the current state's input but also what they have perceived previously in time. Through a loop, the information is passed from one step to another, thus the output at step t is influenced by the decision taken one step before, at timestamp $t-1$. However RNN suffer from short-term memory and can not represent accurately sequences



(a) **The LSTM model architecture**. Both the encoder and the decoder are composed of one LSTM layer. The size of the encoder's input layer is given by the length of the dataset's alphabet. The size of the decoder's input layer is equal to the length of the longest target sequence. The output of the decoder is passed to a dense layer with a softmax activation.



(b) **The LSTM stacked model architecture**. The encoder is composed of three LSTM layers. The output of the third layer of the encoder is passed to both LSTM layers of the decoder. The result is fed to a dense layer with a softmax activation.



(c) **The CNN model architecture**. The encoder and the decoder contain an input layer and three convolutional layers. The outputs from both the encoder and the decoder are passed to an attention module. The obtained output is concatenated again with the decoder's output and fed to two convolutional layers. A fully connected layer with a softmax activation is applied to compute the final output of the network.

Fig. 2: Sequence to sequence model architectures

with long term temporal dependencies. In this situations more complex RNN are preferred. The Long-Short Term Memory (LSTM) [12] units solve the RNN weakness using a three-gated mechanism which controls the amount of information passed to the next step, based on relevance. The input gate selects the information to be stored for the current step, the forget gate discards the unnecessary content, while the output gate will provide the activation to the final output of the LSTM cell.

Convolutional Neural Networks. As a subset of the deep learning algorithms, the Convolutional Neural Networks (CNN) were first used in image processing tasks. Their name is inspired from Latin (*convolvere* = *"to roll together"*) and Mathematics (the convolution is an integral measurement of how two functions' shapes change as they interact). A CNN is composed of successive groups of *convolutional* and *pooling* layers followed by one or more *fully connected* layers at the end. The convolutional layer is responsible for computing the product of two matrixes: one (the *kernel*) represents the set of learnable parameters of the network, the other expresses the restricted portion of the processed

data. The convolutional layer's result is a feature matrix (called *feature map*) with fewer parameters than the original input. The pooling layer (or the subsampling layer) decreases the problem complexity even more as it extracts the dominant characteristics from the feature map. This output is flattened and passed to a fully connected layer. Then, the backpropagation is applied and training continues for several epochs.

The Lemmatizer's architectures. For the sequence-to-sequence Romanian lemmatization task we propose three different neural network architectures. All the systems were trained for 300 epochs. The code was implemented in Python using the functional API of Keras toolkit[5] with a TensorFlow backend[6] and ran on a system with 4 GPUs (Nvidia GeForce RTX 2080 Ti, 11GB memory).

In the first experiments both the encoder and the decoder contain a single LSTM layer. Based on initial tests, the batch size was set to 512 and the latent dimension of the hidden layers to 256. Figures 2a illustrates the system's architecture. Aiming to improve the system accuracy, the encoder and the decoder were enriched with one or two additional LSTM layers, resulting a stacked LSTM hierarchy described in Figure 2b. Based on initial tests, the new system was trained using a batch size of 256 and a latent dimension of 128.

In a last scenario a convolutional architecture with attention module was implemented. Both the encoder and the decoder are composed of 3 convolutional layers with 128 feature map and use the ReLU activation function. The decoder is followed by an attention structure with a softmax activation. The decoder's output is passed through 2 additional convolutional layers with a softmax activation. A final dense layer provides the system's output. Figure 2c describes the network's architecture. A batch size of 512 was chosen after preliminary tests.

5. Experimental setup

Datasets. Two different datasets were used during training. The first dataset is the Romanian Explicative Dictionary[7](ID: **DEX**) which contains 1.158.194 words, each associated with its lemma and part of speech tag. The words are clustered in six major categories based on the part of speech: nouns, adjectives, verbs, pronouns, invariables (adverbs, proper names) and unique forms (interjections, archaic words, Latin names). Table 4 contains the distribution of samples for each POS category.

Table 4: The distribution of samples for each POS category for DEX dataset (left) and CoRoLa dataset (right).

POS - DEX dataset	Abreviation	Percentage
Noun	n	41.28
Adjective	a	28.93
Verb	v	28.00
Unique form	u	1.11
Invariant	i	0.57
Pronoun	p	0.10

POS - CoRoLa dataset	Abbreviation	Percentage
Noun	n	50.08
Verb	v	26.77
Adjective	a	19.00
Adverb	r	2.27
Pronoun	p	0.54
Apposition	s	0.32
Numeral	m	0.30
Conjunction	c	0.15
Hyphen	@	0.18
Abbreviation	y	0.17
Determiner	d	0.16
Article	t	0.05
Particle	q	0.01

The second dataset is the CoRoLa[8] corpus [16] which contains texts from different functional styles: belletristic, scientific, publicistic, official. In this work the belletristic subset was chosen. It contains 51043 sentences with approx. 1 million tokens and 67.460 unique words. Each token belongs to one of the thirteen POS categories (Table 4): noun, verb, adjective, adverb, pronoun, apposition, numeral, conjunction, hyphen, abbreviation, determiner, article and particle.

[5] https://keras.io/
[6] https://www.tensorflow.org/
[7] https://dexonline.ro/
[8] http://CoRoLa.racai.ro/

Several pre-processing steps were performed and include the following operations: convert text to lowercase, strip the digits and punctuation, split sentences into words (CoRoLa dataset), create pairs of input (the word) - target (corresponding lemma) sequences, append a start-marker ("\t") and an end-marker ("\n") to the target word.

Both datasets were subsequently split into disjoint training (80%) and testing (20%) sets (Table 5), each of them being individually shuffled before the splitting.

Table 5: Datasets description

Dataset	Number of samples		
	Total	Training	Testing
DEX	1.158.194	926.556	231.638
Corolla Belletristic - trigrams	753.490	602.792	150.698
Corolla Belletristic - one word	76.417	61.134	15.283

Data Input Type. The neural networks were fed with pairs of (word-lemma). The input data is one-hot encoded, thus the words become bidimensional MxN sparse matrixes, where M is equal to the longest word's length and N is the set of characters that forms all the sequences (number of letters which forms the dataset's alphabet).

First, the systems were forced to learn as much information as possible only from simple pairs of (word and corresponding lemma) (*Input type: word*). Then, the part of speech (POS) for every single word from DEX dataset was appended (*Input type: word + POS*). During training, the input consists of the word and the paired POS, while the system predicts the corresponding lemma: (word|POS → lemma).

In the same idea of adding relevant contextual information, for each word, a context was added by using a window size of three neighboring words to obtain trigrams (*Input type: trigrams*). Only the belletristic subset of CoRoLa corpus could be used, as DEX Dataset contains only isolated words.

For the trigrams scenario, two strategies have been tested. First, we asked the systems to predict the lemmas for the entire sequence of three words. By analysing the results, we observed that in most of the cases the lemma was correctly predicted for only two out of the three words from trigram. Consequently, the systems were also trained to predict the lemma only for the middle word form the trigram (*Input type: trigrams middle*).

6. Results and Discussions

Each architecture described in Figure 2 is trained using each dataset individually, with and without additional POS annotation. For the CoRoLa dataset, the trigram scenario was also analysed, thus, it resulted 18 systems trained using data from CoRoLa (see Table 8) and 6 systems trained over DEX dataset (see Table 6). All 24 systems were evaluated with the *classification accuracy metric*, which is expressed as the ratio between the correct predicted items out of the total samples. The accuracy was measured at different levels: trigram (CoRoLa subset), word and character level (for both datasets). In order to solve the ambiguity problem of words with multiple lemmas, a dictionary of accepted lemmas was built for each dataset. During evaluation, if the predicted lemma belongs to the dictionary of the given word's lemmas, then it is considered to be correct. A dictionary of lemmas is necessary even when POS context is added, as ambiguity can exist within the same POS class, when no additional morphological information is given (genre, case or verb tense).

The results for the DEX dataset are illustrated in Table 6. The one layer LSTM-based architecture achieved the highest accuracy rate at both word and character levels. When additional POS information was provided, the system's accuracy increased by 3.39 % at word level and by 2.14 % at character level.

Unlike the DEX dataset, for the CoRoLa belletristic subset, the system using CNN layers performs better at word and trigrams level (Tables 7 and 8). At character level, the stacked LSTM network achieves the best results. One explanation may be the amount of training data: DEX contains over 900.000 samples compared to only 60.000 samples in the CoRoLa subset. Even if only a small amount of data is available, the CNNs are faster as they are using attention layers and kernels to simulate the context and the recurrence of the data. On the other hand, the LSTM architecture not only fails to predict the correct lemmas, but it also outputs more lemmas than necessary. For instance, in 13% of the trigrams cases, the LSTM-based system predicts the appropriate lemmas for the first and the second word from

the input sequence, while the rest of the output is series of arbitrary lemmas. This means that the end-sequence marker (”\n”) is not correctly predicted. The POS additional information helps all the networks to learn the context from the data, thus the accuracy rate increases by almost 5%.

Although the results are not directly comparable due to different datasets, versioning or experimental details, our proposed architectures obtained higher accuracy than the systems described in Table 1. Thus, for DEX dataset, our LSTM-based architecture, trained using the POS as additional input information (Table 6 system 4), outperforms all the lemmatizers from section 2, while, in the case of CoRoLa dataset, the trigram CNN-based architectures (Table 8) surpass the previously analysed systems (Table 1).

Table 6: Network parameters and accuracy for **DEX** dataset. Best results are marked in bold. At word level, the columns *with ambig. lemma* and *without ambig. lemma* refer to the same target data but check the dictionary created for the words with multiple lemmas.

No.	Input type	Architecture	Latent dimension	Batch size	Accuracy		
					word level		char level
					with ambig. lemma	without ambig. lemma	
1	word	LSTM	256	512	88.20	**95.93**	**97.29**
2		LSTM_stacked	128	256	88.30	94.64	97.17
3		CNN	128	512	90.36	95.83	92.82
4	word + POS	LSTM	256	512	98.10	**99.32**	**99.43**
5		LSTM_stacked	128	256	97.05	98.07	99.12
6		CNN	128	512	97.39	98.36	98.40

Table 7: Network parameters and accuracy for **CoRoLa** dataset, **with one word as input**. Best results are marked in bold. The columns *with ambig. lemma* and *without ambig. lemma* refer to the same target data but check the dictionary created for the words with multiple lemmas.

No.	Input type	Architecture	Latent dimension	Batch size	Accuracy		
					word level		char level
					with ambig. lemma	without ambig. lemma	
7	word	LSTM	256	512	68.03	75.00	88.38
8		LSTM_stacked	128	256	72.08	79.64	**90.68**
9		CNN	128	512	78.51	**86.07**	84.17
10	word + POS	LSTM	128	512	76.77	77.36	89.76
11		LSTM_stacked	128	256	86.55	87.22	**95.05**
12		CNN	128	512	90.72	**91.35**	94.23

7. Conclusions

In this paper 24 systems based on Deep Neural Networks have been analyzed in the context or Romanian lemmatization. The systems were trained on labelled pairs of words and the corresponding lemmas, using at most the part-of-speech tag as morphological information. The input data was one-hot encoded and then passed through encoder-decoder-based architectures, within a sequence-to-sequence approach. The scope of this study was to use as few lexical input information as possible, as the analysed language offers few corpora with completed annotated texts. Apart from fine tuning the network's hyperparameters during training or enriching the input text's metadata, humans can not interfere to improve the learning process in this end-to-end scenario.

As future work, inspired by other studies in the lemmatization task, more types of encoding, such as word embeddings or contextualized embeddings (BERT), will be investigated. In order to leverage the learning process, the

Table 8: Network parameters and accuracy for **CoRoLa** dataset, **with trigrams as input**. Best results are marked in bold. The columns *with ambig. lemma* and *without ambig. lemma* refer to the same target data but check the dictionary created for the words with multiple lemmas.

No.	Input type	Architecture	Latent dimension	Batch size	Accuracy				
					trigram level		word level		char level
					with ambig.	without ambig.	with ambig. lemma	without ambig. lemma	
13	trigrams (middle)	LSTM	256	512	88.69	92.49	97.17	98.58	96.97
14		CNN	128	512	89.56	94.15	98.01	**99.69**	94.82
15		stacked_LSTM	128	256	75.79	79.76	92.12	93.77	91.70
16	trigrams (middle)	LSTM	256	512	N/A	N/A	93.86	95.21	93.28
17		CNN	128	1024			97.07	**98.32**	**96.74**
18		stacked_LSTM	128	256			96.61	95.28	94.86
19	trigrams + POS	LSTM	256	512	62.48	62.84	85.15	85.33	85.05
20		CNN	128	512	95.86	96.49	98.87	**99.09**	97.54
21		stacked_LSTM	128	256	93.31	93.88	97.71	97.91	**97.62**
22	trigrams + POS (middle)	LSTM	256	512	N/A	N/A	98.07	98.07	98.43
23		CNN	128	1024			98.52	98.52	98.11
24		stacked_LSTM	128	256			98.83	98.83	**98.78**

input data encapsulated the lexical and semantic context using n-grams with a sliding window of three neighboring words. Other words window sizes (of five, seven, etc.) will be investigated and their input within the systems will be analysed. Besides CNN and LSTM layers, other types of neural networks will also be explored, such as bidirectional LSTMs, GRU, or only attention based architectures, which are already frequently applied in other text processing tasks. Although the transformer architectures gained popularity in solving NLP tasks for English, Mandarin or other rich resourced languages, we have to analyze the impact on the training process of the limited amount of data if we want to apply these systems to Romanian language.

8. Acknowledgments

This work was supported by a grant of the Romanian Ministry of Research and Innovation, PCCDI – UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0818/73, within PNCDI III.

References

[1] Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., Vollgraf, R., 2019. Flair: An easy-to-use framework for state-of-the-art nlp, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), Association for Computational Linguistics, Minneapolis, Minnesota. pp. 54–59.

[2] Badaro, G., Jundi, H., Hajj, H., El-Hajj, W., Habash, N., 2018. Arsel: A large scale arabic sentiment and emotion lexicon. OSACT 3, 26.

[3] Boroş, T., 2013. A unified lexical processing framework based on the margin infused relaxed algorithm. a case study on the romanian language, in: Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013, INCOMA Ltd. Shoumen, Hissar, Bulgaria. pp. 91–97.

[4] Boroş, T., Dumitrescu, S.D., Burtica, R., 2018. Nlp-cube: End-to-end raw text processing with neural networks, in: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Association for Computational Linguistics, Brussels, Belgium. pp. 171–179.

[5] Boros, T., Dumitrescu, S.D., Pais, V., 2018. Tools and resources for romanian text-to-speech and speech-to-text applications, in: Proceedings of the International Conference on Human-Computer Interaction RoCHI 2018, MATRIX ROM, Bucharest, Romania. pp. 46–53.

[6] Chakrabarty, A., Pandit, O.A., Garain, U., 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), Association for Computational Linguistics, Vancouver, Canada. pp. 1481–1491.

[7] Chrupała, G., Dinu, G., Van Genabith, J., 2008. Learning morphology with morfette, in: Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), European Language Resources Association (ELRA), Marrakech, Morocco. pp. 2362–2367.

[8] Crammer, K., Singer, Y., 2003. Ultraconservative online algorithms for multiclass problems. Journal of Machine Learning Research 3, 951–991.
[9] Desmet, B., Hoste, V., 2013. Emotion detection in suicide notes. Expert Systems with Applications 40, 6351–6358.
[10] Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Association for Computational Linguistics, Minneapolis, Minnesota. p. 4171–4186.
[11] El-Shishtawy, T., El-Ghannam, F., 2014. A lemma based evaluator for semitic language text summarization systems. arXiv:1403.5596.
[12] Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780.
[13] Irimia, E., 2009. Ebmt experiments for the english-romanian language pair, in: Recent Advances in Intelligent Information Systems, Academic publishing house EXIT, Warsaw, Poland. pp. 91–102.
[14] Kanerva, J., Ginter, F., Salakoski, T., 2019. Universal lemmatizer: A sequence to sequence model for lemmatizing universal dependencies treebanks. arXiv:1902.00972.
[15] Manolache, C., Cucu, H., Burileanu, C., 2019. Lemma-based dynamic time warping search for keyword spotting applications in romanian, in: 2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD), IEEE, Timisoara, Romania. pp. 1–9.
[16] Mititelu, V.B., Irimia, E., Tufis, D., 2014. Corola - the reference corpus of contemporary romanian language, in: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), European Language Resources Association (ELRA), Reykjavik, Icelan. pp. 1235–1239.
[17] Myers, E.W., 1986. Ano (nd) difference algorithm and its variations. Algorithmica 1, 251–266.
[18] Pak, A., Bernhard, D., Paroubek, P., Grouin, C., 2012. A combined approach to emotion detection in suicide notes. Biomedical informatics insights 5, BII–S8969.
[19] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L., 2018. Deep contextualized word representations. arXiv:1802.05365.
[20] Pouget, M., Nahorna, O., Hueber, T., Bailly, G., 2016. Adaptive latency for part-of-speech tagging in incremental text-to-speech synthesis, in: Proceedings of the 17th Annual Conference of the International Speech Communication Association (INTERSPEECH 2016): Understanding Speech Processing in Humans and Machines, International Speech Communication Association, San Francisco, USA. pp. 2846–2850.
[21] Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D., 2020. Stanza: A python natural language processing toolkit for many human languages. arXiv:2003.07082.
[22] Rheault, L., Beelen, K., Cochrane, C., Hirst, G., 2016. Measuring emotion in parliamentary debates with automated textual analysis. PloS one 11, e0168843.
[23] Shen, J., Pang, R., Weiss, R.J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., Saurous, R.A., Agiomvrgiannakis, Y., Wu, Y., 2018. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Calgary, AB, Canada. pp. 4779–4783.
[24] Skorkovská, L., 2012. Application of lemmatization and summarization methods in topic identification module for large scale language modeling data filtering, in: International Conference on Text, Speech and Dialogue, Springer, Berlin, Heidelberg. pp. 191–198.
[25] Straka, M., Straková, J., Hajič, J., 2019. Evaluating contextualized embeddings on 54 languages in pos tagging, lemmatization and dependency parsing. arXiv:1908.07448.
[26] Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to sequence learning with neural networks, in: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, MIT Press, Cambridge, MA, USA. p. 3104–3112.
[27] Wang, Y., Stanton, D., Zhang, Y., Skerry-Ryan, R., Battenberg, E., Shor, J., Xiao, Y., Ren, F., Jia, Y., Saurous, R.A., 2018. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. arXiv:1803.09017.
[28] Wu, Z., Watts, O., King, S., 2016. Merlin: An open source neural network speech synthesis system., in: proceedings of the 9th International Speech Communication Association (ISCA) Speech Synthesis Workshop: SSW 2016, International Speech Communication Association (ISCA), Sunnyvale, United States. pp. 202–207.
[29] Yildiz, E., Tantuğ, A.C., 2019. Morpheus: A neural network for jointly learning contextual lemmatization and morphological tagging, in: Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology, pp. 25–34.
[30] Zine, O., Meziane, A., Boudchiche, M., 2017. Towards a high-quality lemma-based text to speech system for the arabic language, in: International Conference on Arabic Language Processing, Springer. Springer, Cham, Fez, Morocco. pp. 53–66.