UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

UNIVERSITATEA
„ALEXANDRU IOAN CUZA“
din IAȘI

D2.15. Implementarea modulului de identificare a stilului de vorbire și nivelului de expresivitate din analiza textului

Aceste rezultate au fost obținute prin finanțare în cadrul Programului PN-III Proiecte complexe realizate în consorții CDI, derulat cu sprijinul MEN – UEFISCDI,
Cod: PN-III-P1-1.2-PCCDI-2017-0818, Contract Nr. 73 PCCDI/2018:

“SINTERO: Tehnologii de realizare a interfețelor om-mașină pentru sinteza text-vorbire cu expresivitate”

Denumirea organizației participante în proiect	Acronim organizație	Tip organizație	Rolul organizației în proiect (Coordonator/partener)
Institutul de Cercetări Pentru Inteligență Artificială “Mihai Drăgănescu”	ICIA	UNI	CO
Universitatea Tehnică din Cluj-Napoca	UTCN	UNI	P1
Universitatea Politehnica din București	UPB	UNI	P2
Universitatea "Alexandru Ioan Cuza" din Iași	UAIC	UNI	P3

MINISTERUL EDUCAȚIEI NAȚIONALE

uefiscdi

Date de identificare proiect

Număr contract:	PN-III-P1-1.2-PCCDI-2017-0818, Nr. 73 PCCDI/2018
Acronim / titlu:	**„SINTERO: Tehnologii de realizare a interfețelor om-mașină pentru sinteza text-vorbire cu expresivitate”**
Titlu livrabil:	**D2.15. Implementarea modulului de identificare a stilului de vorbire și nivelului de expresivitate din analiza textului**
Termen:	**August 2019**
Editor:	**Mircea Giurgiu (Universitatea Tehnică din Cluj-Napoca)**
Adresa de eMail editor:	**Mircea.Giurgiu@com.utcluj.ro**
Autori, în ordine alfabetică:	**Mircea Giurgiu, Adriana Stan**
Ofițer de proiect:	**Cristian STROE**

Rezumat:

Pentru un control mai bun al sistemelor de sinteză și pentru ca acestea să poată reda textul introdus într-o manieră cât mai apropiată de vocea naturală, este util ca datele de intrare (textul) să poată fi clasificat automat în funcție de stilul pe care acesta îl conține.

În cadrul acestui raport vom prezenta două metode de detecție a stilului textului bazate pe metoda Latent Dirichlet Allocation (LDA), respectiv pe rețele neurale convoluționale multistrat. De asemenea pentru a îmbunătăți sistemul de detecție a stilului, vom prezenta și două module pentru restaurarea diacriticelor și determinarea părții de vorbire a cuvintelor. Cele două module sunt incluse în fluxul de procesare la intrarea sistemului de clasificare a textului.

Cuprins

1. Introducere

Pentru un control mai bun al sistemelor de sinteză și pentru ca acestea să poată reda textul introdus într-o manieră cât mai apropiată de vocea naturală, este util ca datele de intrare (textul) să poată fi clasificat automat în funcție de stilul pe care acesta îl conține.

În cadrul acestui raport vom prezenta două metode de detecție a stilului textului bazate pe metoda Latent Dirichlet Allocation (LDA), respectiv pe rețele neurale convoluționale multistrat. De asemenea pentru a îmbunătăți sistemul de detecție a stilului, vom prezenta și două module pentru restaurarea diacriticelor și determinarea părții de vorbire a cuvintelor. Cele două module sunt incluse în fluxul de procesare la intrarea sistemului de clasificare a textului.

2. Clasificarea automată a textului folosind Latent Dirichlet Allocation

În prima etapă a cercetărilor s-a dezvoltat o metodă de clasificare automată a stilului de vorbire din text bazată pe tehnici tradiționale de prelucrare a limbajului, în speță metoda LDA (Latent Dirichlet Allocation) folosită deja cu succes în sisteme de identificare automată a genului scrierii și mai ales în sisteme de clasificare automată a topicului discursului. Astfel, am plecat de la ipoteza că fiecare stil de exprimare scrisă poate fi modelat sub forma unui model echivalent de topic. Astfel, LDA ca model probabilist este capabil să modeleze în mod ierarhic fiecare stil de exprimare ca o combinație finită de probabilități de stiluri de exprimare, din cele disponibile în mod latent în setul de antrenare. Această modelare este potrivită, mai ales având în vedere spectrul larg de posibilități de exprimare în diferitele stiluri de vorbire.

Corpusul de text folosit este obținut de la Coordonatorul proiectului (ICIA) și conține text adnotat pe diferite nivele de adnotare (lema, părțile de vorbire, etc) pentru 5 categorii de stiluri de exprimare: publicistic, politic, memorialistic, juridic, beletristic.

Nume corpus	Numărul documentelor	Numărul de cuvinte
Publicistic.txt	*848*	*325191*
Politic.txt	*8*	*374680*
Memorialistic.txt	*12*	*315213*
JuridicAdministrativ.txt	*153*	*337143*
Beletristic.txt	*180*	*251927*

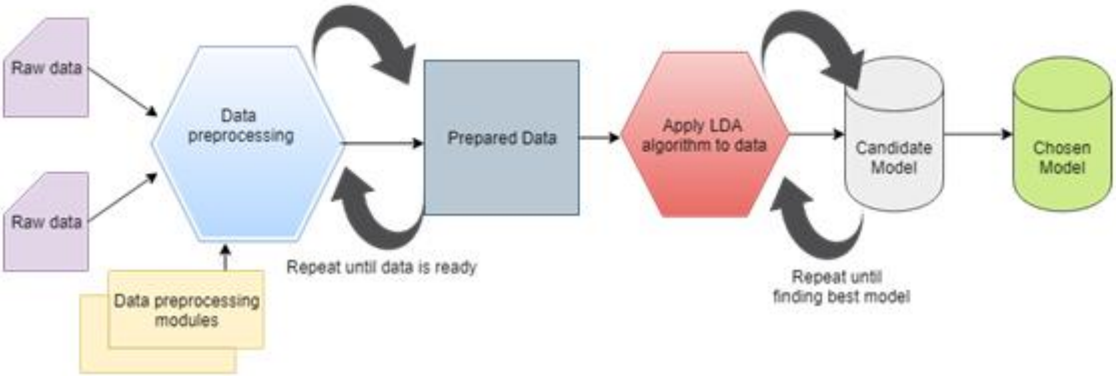
Raw data
Raw data
Data preprocessing
Data preprocessing modules
Prepared Data
Repeat until data is ready
Apply LDA algorithm to data
Candidate Model
Chosen Model
Repeat until finding best model

Fig. 1. Fluxul de prelucrări

Implementarea fluxului de prelucrări include: (1) eliminarea unor cuvinte de tipul “stop words” (cuvinte fără relevanță în clasificarea stilului de exprimare, de exemplu ‚cu’, ‘de’, ‘pe’, etc., preluate dintr-o listă generată de utilizator și customizabilă pentru a experimenta efectul eliminării unor cuvinte specifice), (2) lematizare – reținerea rădăcinii cuvintelor, cu scopul de a limita numărul total de cuvinte, în condițiile în care semantica nu este afectată, (3) segmentare text la nivel de propoziție, (4) eliminare caractere speciale, (5) creare de n-grame (eg. director-general), (6) creare dicționar – etapa are rolul de a converti textul în codificare numerică, (6) creare modele LDA, (7) alegerea modelelor folosind ca și criterii de selecție perplexitatea și scorul de coerentă, (8) reprezentare vizuală a rezultatelor.

Rezultatele sunt vizibile sub formă grafică, sub forma probabilităților de clasificare a unui text necunoscut către unul dintre stilurile din corpusul de antrenare (mai jos grafic rezultat din 4 stiluri), precum și a scorului de coerență ale modelelor (ca măsură a gradului de separabilitate ale modelelor).

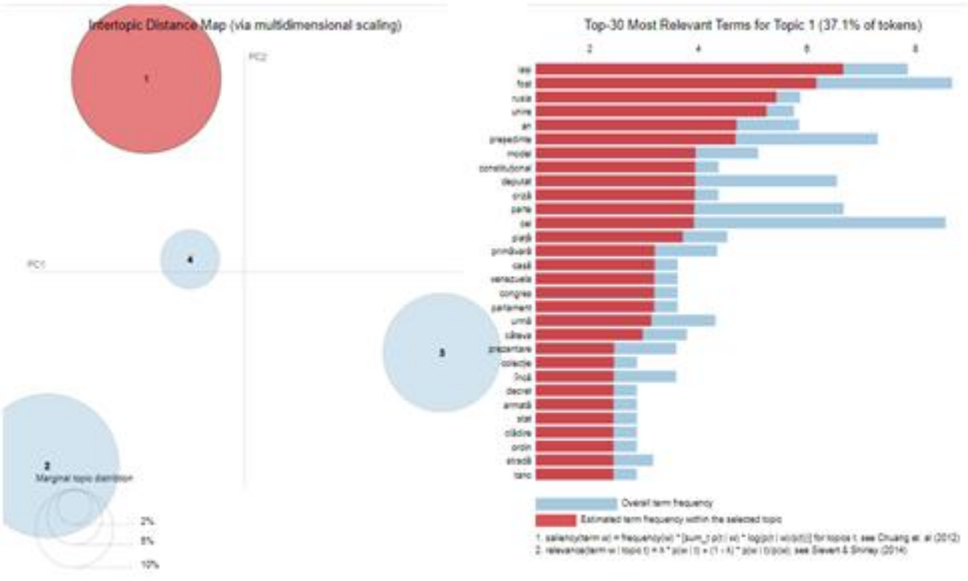

Fig. 2. Grafice pentru gradul de separabilitate a celor 4 modele și cuvintele definitorii

3. Clasificarea automată a textului folosind rețele convoluționale multistrat

Odată dezvoltarea tehnologiilor de calcul și a capabilităților sistemelor bazate pe procesoare grafice, algoritmii ce utilizează rețele neuronale multistrat au prezentat un salt în majoritatea domeniilor în care aceștia au fost aplicați: procesarea limbajului natural, sisteme de procesare a vocii, viziune computerizată, etc. Astfel că, utilizarea lor în clasificarea stilului funcțional al unui text merită analizată.

În ceea ce privește clasificarea textului folosind rețele neuronale multistrat, există o serie de studii recente privind, în special, aplicarea rețelelor convoluționale uni-dimensionale pentru a rezolva această problemă. Dintre acestea, cele mai des utilizate arhitecturi sunt cele de tip sequence-to-sequence (Gehrig et al., 2017). Acest tip de arhitectură permite utilizarea secvențelor de dimensiuni diferite atât în cadrul datelor de intrare ale rețelei, cât și la cele de ieșire. Pentru clasificarea textului cu ajutorul rețelelor convoluționale (Kim, 2014) folosește reprezentări vectoriale ale cuvintelor învățate dintr-un alt set de date, combinate cu o rețea neuronală cu arhitectură relativ simplă și obține rezultate ce depășeau alte studii de la momentul respectiv. (Lee et Dernoncourt, 2016) evaluează utilizarea rețelelor convoluționale și recurente pentru clasificarea textului și includ suplimentar informații despre secvențialitatea textului. Aceasta înseamnă că pentru textul curent, se va lua în considerare și textul clasificat anterior, dând o oarecare secvențialitate procesului de clasificare. O arhitectură bazată pe rețele convoluționale, puțin mai complexă este descrisă în (Wang et al., 2017). Aceasta include și un set de descriptori impliciți și expliciți ai textului de intrare prin conceptualizarea unor termeni utilizând o taxonomie derivată dintr-un corpus extins de date. Descriptorii astfel obținuți sunt comparați o reprezentare vectorială des utilizată denumită GloVE (Pennington et al., 2014). Pe baza acestor descriptori, rețeaua reușește să clasifice texte de dimensiuni reduse cu o acuratețe de până la 93%.

3.1. Descrierea metodei

Pornind de la rezultatele studiilor anterioare, am dorit să aplicăm arhitecturi de rețele convoluționale asupra textelor în limba română. Un prim pas al acestui experiment se referă la preprocesarea textului de intrare, prin:

- segmentare la nivel de cuvânt;
- eliminarea semnelor de punctuație;
- filtrarea simbolurilor alfabetice;
- conversia la litere mici;
- vectorizarea cuvintelor prin atribuirea unui index de valoare întreagă;
- completarea cu valori de zero a frazelor de lungime mai mică decât cea prestabilită.

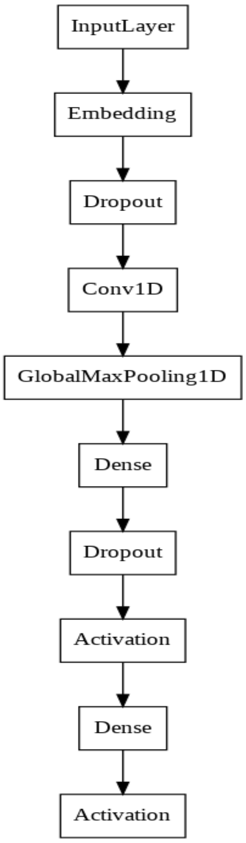

Fig. 2. Arhitectura rețelei convoluționale utilizată pentru clasificarea textului.

Textul astfel procesat a fost apoi trecut printr-o rețea neuronală. Arhitectura rețelei[1][2] este inspirată din (Kim, 2014), se bazează pe straturi de tip convoluțional și este prezentată în Figura 1. Rețeaua include un prim strat de reprezentare vectorială (en. *embedding),* urmat de un strat de tip dropout (eliminare a unui procent de neuroni), un strat de tip convoluțional cu filtre de dimensiune 3, 4 sau 5 cu activare ReLU și pas 1. Ieșirea stratului convoluțional este procesată de un strat de tip max pooling, un strat complet conectat și unul de tip dropout cu activare ReLU. Clasificarea finală este dată de un strat complet conectat de dimensiune 5 (corespunzător numărului de stiluri) și funcție de activare de tip softmax.

3.2. Rezultate

Metoda descrisă anterior a fost aplicată asupra unui set de date text extrase din corpusul Corola al Institutului de Inteligență Artificială al Academiei Română din București (Mititelu et al., 2018). Setul de date conține text în stilurile: *beletristic, științific, publicistic, memorialistic și juridic*. Pentru fiecare subset am avut la dispoziție aproximativ 1 milion de tokeni (cuvinte), organizate în 40,000 de fraze. Media numărului de cuvinte dintr-o frază este de 20.

Au fost evaluate mai multe scenarii pentru această rețea prin varierea cantității de date folosită pentru antrenarea rețelei, a dimensiunii stratului convoluțional și a numărului de epoci de antrenare. Pentru fiecare dintre aceste combinații s-au reținut din datele de antrenare 20% pentru testare și 10% pentru validare. Rezultatele metodei sunt prezentate în Tabelul 1.

Codul ce permite rularea modelului cel mai bun este disponibil la adresa: https://github.com/speech-utcluj/romanian-text-classification-cnn.

Tabel 1.. Rezultatele metodei de clasificare a textului folosind rețele neuronale convoluționale

Nr.	Număr propoziții antrenare	Dimensiune conv.	Număr epoci	Acuratețe
1	5*3000	512	25	93.45%
2.			50	93.37%
3.		1024	25	92.77%
4			50	93.46%
5.	5*1000	512	25	91.83%
6.			50	91.81%
7.		1024	25	91.37%
8.			50	91.63%
9.	5*8000	512	25	92.69%
10.			50	92.41%
11.		1024	25	92.72%
12.			50	92.28%
13.	5*38000	512	25	90.10%
14.		1024	25	90.44%

Se poate observa că folosind această arhitectură, algoritmul este capabil să clasifice datele cu o acuratețe relativ mare și poate fi astfel folosit în pașii următori ai sintezei text-vorbire. Chiar și cu date puține (1000 de propoziții/stil) rezultatele algoritmului sunt de aproximativ 91%. Creșterea numărului de epoci nu influențează rezultatele, iar rezultale folosind numărul maxim de propoziții din fiecare stil ajung la o acuratețe de mai mare de 90%.

În ceea ce privește dezvoltările ulterioare, ne dorim să analizăm posibilitatea reducerii dimensiunii textului de intrare la propoziții mai scurte și eventual atașarea în cadrul rețelei a unor informații de ordin lingvistic (de ex. parte de vorbire, parsare sintactică, etc.).

[1] http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/

[2] https://richliao.github.io/supervised/classification/2016/11/26/textclassifier-convolutional/

4. Restaurarea automată a diacriticelor

Lipsa diacriticelor este predominantă în textele electronice datorită faptului că utilizatorii fie nu folosesc editoare de text adecvate, fie nu au abilități de editare electronică suficiente. Ca urmare, utilizarea datelor colectate din surse online este de cele mai multe ori obstrucționată de acest fapt. Dezvoltarea unui modul de restaurare a diacriticelor în vederea unei mai bune exploatări a numeroaselor resurse disponibile este astfel absolut necesară. În procesul de clasificare a stilului unui text, am introdus astfel un pas de preprocesare a textului ce include un sistem automat de restaurare de diacritice.

O descriere completă a sistemului, experimentelor și rezultatelor privind restaurarea automată a diacriticelor a fost prezentată în cadrul conferinței 15th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP 2019), septembrie, 2019, Cluj-Napoca, Romania. Articolul este anexat la acest raport (Anexa 1).

5. Detecția automată a părții de vorbire

Pentru a putea clasifica textul în funcție de stilul conținut, este util în anumite cazuri să se realizeze o dezambiguare a înțelesului unui cuvânt. Această dezambiguare este realizată de cele mai multe ori folosind partea de vorbire a cuvântului. Astfel că, am derulat o serie de experimente privind utilizarea unor arhitecturi de rețele neuronale pentru a prezice această informație lingvistică.

O descriere completă a metodei, a experimentelor și rezultatelor privind detecția automată a părții de vorbire la nivel de cuvânt a fost prezentată în cadrul conferinței 15th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP 2019), septembrie, 2019, Cluj-Napoca, Romania. Articolul este anexat la acest raport (Anexa 2).

6. Analiză comparativă a trăsăturilor utile în transcrierea fonetică

Deși nu are implicații directe în evaluarea stilului unui text, transcrierea fonetică reprezintă un pas esențial în prelucrarea de voce și limbaj natural. Astfel că, în conformitate cu experimentele anterioare, am dorit să evaluăm modul în care tipul de reprezentare a textului și informația lingvistică suplimentară introdusă în datele de intrare ale sistemului de transcriere fonetică influențează performanțele acestuia.

O descriere completă a experimentelor și rezultatelor privind detecția automată a părții de vorbire la nivel de cuvânt a fost prezentată în cadrul conferinței 10th IEEE International Conference on Speech Technology and Human-Computer Dialogue (SpeD), octombrie 2019, Timișoara, Romania. Articolul este anexat la acest raport (Anexa 3).

7. Concluzii

În acest raport am prezentat o serie de experimente și rezultate referitoare la metode automate de clasificare a textului în funcție de stilul funcțional al său, precum și metode de preprocesare ce pot îmbunătăți rezultatele acestor metode: restaurarea automată a diacriticelor, precum și detecția automată a părții de vorbire. În etapele următoare se va urmări utilizarea rezultatelor raportate pentru îmbunătățirea sistemului de sinteză în limba română, precum și expresivitatea acestuia.

8. Bibliografie

(Gehrig et al., 2019)	Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin, ”Convolutional Sequence to Sequence Learning”, arXiv:1705.03122
(Kim, 2014)	Yoon Kim, ”Convolutional Neural Networks for Sentence Classification”, EMNLP, 2014, arXiv:1408.5882
(Lee et Dernoncourt, 2016)	Ji Young Lee, Franck Dernoncourt, ”Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks”, Proceedings of the 2016 Conference of the North American Chapter of the Association for

	Computational Linguistics: Human Language Technologies, 2016
(Mititelu et al., 2018)	Verginica Barbu Mititelu, Dan Tufiș, Elena Irimia, The Reference Corpus of the Contemporary Romanian Language (CoRoLa), in Proceedings of LREC 2018, Japan, p.1178-1185.
(Pennington et al., 2014)	Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In EMNLP, pages 1532–1543, 2014.
(Wang et al., 2017)	Wang, J., Wang, Z., Zhang, D., & Yan, J. (2017, August). Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In IJCAI (pp. 2915-2921).
(Zhang et Zhong, 2016)	Zhang, Heng, and Guoqiang Zhong. "Improving short text classification by learning vector representations of both words and hidden topics." Knowledge-Based Systems 102 (2016): 76-86.

ANEXA 1. Articol: Maria Nutu, Beáta Lőrincz, Adriana Stan, "Deep Learning for Automatic Diacritics Restoration in Romanian", In Proceedings of the IEEE 15th International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, 2019.

ANEXA 2. Articol: Beáta Lőrincz, Maria Nuțu, Adriana Stan, "Romanian Part of Speech Tagging using LSTM Networks", ***In Proceedings of the IEEE 15th International Conference on Intelligent Computer Communication and Processing*****, Cluj-Napoca, Romania, 2019.**

ANEXA 3. Articol: Adriana Stan, "Input Encoding for Sequence-to-Sequence Learning of Romanian Grapheme-to-Phoneme Conversion", In Proceedings of the 10th IEEE International Conference on Speech Technology and Human-Computer Dialogue(SpeD), Timișoara, Romania, 2019.

Deep Learning for Automatic Diacritics Restoration in Romanian

Maria Nuţu[a,b], Beáta Lőrincz[a,b], Adriana Stan[a]
[a]Communications Department, Technical University of Cluj-Napoca, Romania
[b]Department of Computer Science, ”Babeş-Bolyai” University, Cluj-Napoca, Romania
{maria.nutu, beata.lorincz, adriana.stan}@com.utcluj.ro

***Abstract*—In this paper we address the issue of automatic diacritics restoration (ADR) for Romanian using deep learning strategies.**

We compare 6 separate architectures with various mixtures of recurrent and convolutional layers. The input consists in sequences of consecutive words stripped of their diacritic symbols. The network's task is to learn to restore the diacritics for the entire sequence. No additional linguistic or semantic information is used as input to the networks.

The best results were obtained with a CNN-based architecture and achieved an accuracy of 97% at word level. At diacritic-level the accuracy of the same architecture is 89%.

***Index Terms*—automatic diacritics restoration, deep neural networks, LSTM, CNN, sequence-to-sequence, Romanian**

I. Introduction

Automatic Diacritics Restoration (ADR) is the process of restoring the diacritic symbols in orthographic texts. The applications of this process are numerous and include: spelling checkers, lexical disambiguation, part-of-speech tagging, natural language understanding, etc. The lack of diacritics is predominant in electronic texts where the user does not use adequate text editing software, or is not technologically proficient so as to use the diacritic symbols specific to his or her native/acquired language.

Most of the European languages contain different sets of diacritic symbols in their alphabets, with the most numerous being in French and Slovak. The set of diacritics used in European languages based on the Latin alphabet are illustrated in Table I.

Romanian uses 5 diacritic letters: *ă, â, î, ş* and *ţ*. Although not all words have alternative spellings with and without diacritics, in some cases, a missing diacritic could completely change a word's meaning (e.g *peste* = over vs. *peşte* = fish), while in other cases, the absence of the appropriate diacritic in the word's ending letter makes it impossible to discern between the definite or indefinite form of a noun (*mamă* = a mother vs. *mama* = the mother).

Tufiş et al. [1] reports that between 25% and 45% of the Romanian words contain diacritics, while in a random French text, only 15% of the words contain diacritic symbols [2]. The diacritic percentage across the European languages is reported in [3].

Motivated by the the relevance of diacritic restoration across various text-based applications, in this work we address the Romanian ADR problem using sequence-to-sequence deep learning architectures based on convolutional and recurrent neural networks.

TABLE I
Diacritics in European Languages with Latin based alphabets

Language	Diacritics	Language	Diacritics
Albanian	ç ë	Italian	á è é ì í ï ò ó ù ú
Basque	ñ ü	Lower Sorbian	ċ č ĕ ł ń ŕ ś š ź ž
Breton	â ê ñ ú ö	Maltese	ċ ġ ż
Catalan	à ç è é í ï ò ó ú ü	Norwegian	å æ ø
Czech	á č é í n̄ ó r̄ s̄ ý z̄	Polish	ą ę ć ł ń ó ’s ’z ż
Danish	å æ ø	Portuguese	â ă ç ê ó ô ŭ ü
Dutch	ë	Romanian	ă â î ş ţ
English	none	Sami	á ï ĉ d- ń ŋ š t- ž
Estonian	ä č ō ö ž	Serbo-Croatian	ć č d- š ž
Faroese	á æ d- ó øú ý	Slovak	á ä č d’ é Í ñ ó ô ŕ š t’ ú ý ž
Finnish	ä å ö š ž	Slovene	č š ž
French	á â æ ç é è ë ê î œ ù û ÿ	Spanish	á é í ó ú ü ñ
Gaelic	á é í ó ú	Swedish	ä å ö
German	ä ö ü ß	Turkish	ç ğ ö ş ü
Hungarian	á é í ó ö ő ú ü ű	Upper Sorbian	ć č ĕ ł ń ó ř š ž
Icelandic	á æ ∂ é í ó ö ú ý	Welsh	ă ĕ ǐ ŏ ŭ ŵ ŷ

The paper is structured as follows: Section II is a brief overview of the state-of-the-art methods used in ADR. Section III outlines the sequence-to-sequence architectures, while Section IV presents the dataset and the tested architectures. The final results are illustrated in Section V. The conclusions and future perspectives are summarized in Section VI.

II. Related work

With the increase in the use of electronic devices across different social and cultural categories, the need for high-quality ADR applications is more prevalent, and so is the number of published scientific studies. Simard [2] employs Hidden Markov Models trained at word level on French texts. For the Vietnamese language, Nguyen et al. [4] combine Adaboost and C4.5 decision tree classifiers with a letter-based feature set in five different strategies: learning from letters, learning from semi-syllables, learning from syllables, learning from words, and learning from bi-grams.

A deep learning approach for diacritics restoration is proposed by Náplava et. al. in [5] and uses Bidirectional Neural Networks combined with a language model. The model was tested for 23 languages, including among others Czech, Slovak and Romanian.

For the Romanian language, in particular, the works of Mihalcea et al. [3], [6] explore instance based learning at letter

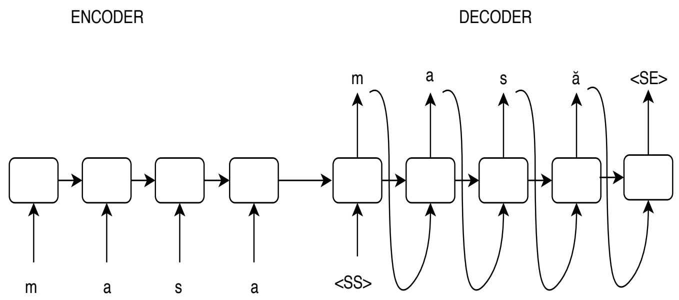

Fig. 1. Sequence-to-sequence flow

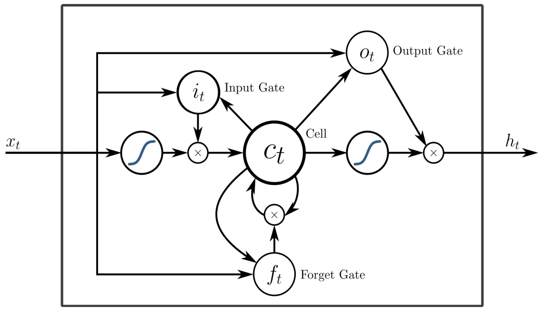

Fig. 2. LSTM memory cell [11]

level, using the Tilburg memory and the C4.5 decision tree classifier, scoring an overall F-measure of 98.30 %.

Tufiş et al. [1] propose a Part-Of-Speech tagger and the use of two lexicons to solve the ambiguity problem in Romanian ADR. An overall accuracy of 97.4 % is achieved at word level.

Ungureanu et. al [7] propose a word classification schema, based on the occurrence of diacritics in each word (words always written with diacritics, words with no diacritics at all and words with different diacritical written pattern - words which change their meaning as diacritics are missing, as shown in Section I). Then these categories are distilled into two dictionaries. During training and testing, the two lexicons are used to improve the ADR results, obtaining an overall F-measure of 99.34%.

In [8] Petrică presents a diacritics restoration system trained on unreliable raw data sets. First, the correctly spelled sections are identified and used as training data for the ADR. Second, the trained ADR is applied to the remaining parts of the initial text.

The previously described approaches use language models and linguistic information extracted from the texts at different levels. In this work, we propose a deep learning approach to solve the ADR problem for Romanian using only character sequences and without any expert linguistic knowledge.

III. Sequence to sequence learning

The sequence-to-sequence (seq2seq) [9] architecture is designed to handle input and output sequences with different lengths. The most common applications for this architecture include automatic machine translation, video captioning, speech recognition and speech synthesis.

Broadly speaking, the seq2seq architecture is formed of two parts: an *encoder* and a *decoder*, each of them being a separate neural network. The encoder is responsible for understanding 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 ”masa” as input and ”masă” as output. The 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.

A. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural network in which the output of the current time step is conditioned on the output of the previous time step. As a result, RNNs are commonly used to model temporal sequences. However, a major problem with vanilla RNNs is that they cannot model sequences in which the temporal dependencies are stretched across multiple time steps.

The solution for this problem is to use more advanced network nodes, in which an internal state of the node can memorize or forget data snippets which are of interest to the current prediction. One such specialized node is the Long Short Term Memory (**LSTM**) cell [10].

A LSTM cell (graphically depicted in Figure 2) contains the following elements:

- forget gate f_t - a neural network (NN) with sigmoid activation
- input gate i_t - a NN with sigmoid activation
- output gate o_t - a NN with sigmoid activation
- hidden state h_t - a vector
- memory state c_t - a vector

The input gate selects what new information to be stored in the current cell at a time step *t*. The forget-gate expresses the amount of information which will be discarded, while the output-gate will provide the activation to the final output of the LSTM block. The hidden state is calculated from the cell state passed through an activation function and element-wise multiplied with the output vector at the time step t.

B. Convolutional Neural Networks

Convolutional Neural Networks (CNN), originally used in image processing, are another type of deep networks largely used for pattern recognition tasks.

A simple CNN architecture contains the following elements:

- a convolutional layer
- a non-linear activation layer
- a pooling (or sub sampling) layer
- a fully connected (softmax) output layer.

The convolutional layer defines a non-linear filter bank (or kernel), which is shifted over the input features using a fixed stride and generates a multi-dimensional feature map, which is processed by a non-linear activation function. The pooling layer reduces the representation of the convolutional layer's output, as well as decreases the memory requirements. In general, the polling layer is placed between the convolutional layers. The features with the highest values (maxpool) are fed into a fully connected layer, whose activations are finally passed into a softmax layer. The output of the softmax function represents the estimated probability distribution over the output labels. In some cases, a normalization layer is stacked on the pooling layer to normalize the data, with mean 0 and variance 1. The normalization step ensures the networks stability.

The characteristics highlighted above make the seq2seq learning a good candidate for the Romanian ADR problem.

IV. Experimental setup

A. Training Data

For training and testing our models, we selected a subset of the CoRoLa text corpus [12]. The subset contains 51.043 sentences with 1 million tokens and 63.194 unique words. The style of the text is belletristic. The corpus is not purposely build for ADR tasks, but can be considered as a reliable source of correctly typed text (i.e. containing the correct diacritics) in Romanian as it was manually annotated at word-level with several linguistic information. We subsequently split the dataset into disjoint training (80%) and testing (20%) sets, each of them being individually shuffled.

A few pre-processing steps were performed and include the following operations:

- convert text to lowercase
- strip the digits and punctuation
- strip the diacritics
- parse the text in trigrams
- create pairs of input-target sequences
- append a start-character (”\t”) and an end-character (”\n”) to the target trigram

An example of a pre-processed sentence is shown in Table II. The obtained input-output pairs are illustrated in Table III.

TABLE II
PRE-PROCESSING EXAMPLE

Initial sentence	”Mă uitasem la ceas, era încă ora 22.00.”
Pre-processed sentence	”ma uitasem la ceas era inca ora”

TABLE III
INPUT-OUTPUT TRIGRAMS FOR A CHOSEN SENTENCE

Input sequence	Target sequence
ma uitasem la	\t mă uitasem la \n
uitasem la ceas	\t uitasem la ceas \n
la ceas era	\t la ceas era \n
ceas era inca	\t ceas era încă \n
era inca ora	\t era încă ora \n

When an unknown input sequence is decoded, we begin with the starting character and use the decoder to predict the next character until the ending character is generated. The trigrams were chosen to represent the context of the current sequence.

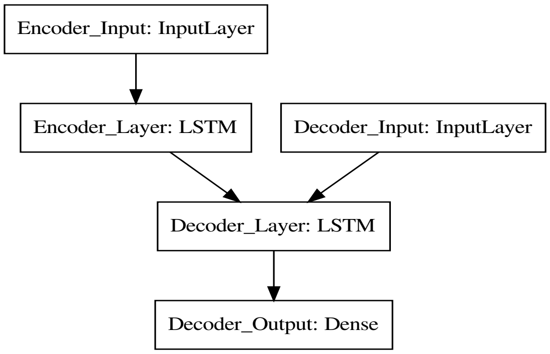

Fig. 3. **seq2seq-LSTM** model architecture

After the pre-processing steps, the train set ended-up containing 616.691 tokens, while the test set contained 162.791 tokens.

B. System architectures

For our initial tests we selected two ADR systems [5], [13] previously applied for Romanian. The systems were retrained using our dataset, but preserving the original parameter values.

Inspired by the architectures described in these two systems, we analyzed four other architectures with various combinations of recurrent and convolutional layers. For implementation, we relied on Keras[1] with TensorFlow[2] as backend. The networks' hyperparameters were tuned using a small development set.

All 6 architectures are described in the following subsections with the previously published works marked with an asterisk (*). All systems were trained over 50 epochs.

*1) One layer LSTMs (**ID: seq2seq_LSTM**):* In the RNN sequence-to-sequence architecture the encoder and decoder both included one LSTM layer. A latent dimension of 128 for both layers and a batch size of 512 were chosen. The input to the encoder and decoder was one-hot encoding at character level. The input of the decoder was also conditioned on the hidden state of the encoder. The output of the decoder LSTM layer is sent to a softmax dense layer with a dimension equal to the length of the one-hot encoded target character set. Figure 3 illustrates the architecture design of the RNN architecture.

*2) Stacked LSTMs (**seq2seq_stacked_*_LSTM**):* In order to improve the results, one additional LSTM layer was added to the encoder. The newly obtained encoder was tested in two different contexts. First, we used a 1 LSTM layer for the decoder (**ID: seq2seq_stacked_1_LSTM**). Then, another LSTM layer was stacked in the decoder (**ID: seq2seq_stacked_2_LSTM**).

[1] https://keras.io/
[2] https://www.tensorflow.org/

The model **seq2seq_stacked_1_LSTM** was trained with a 256 latent dimension and 128 batch size. For model **seq2seq_stacked_2_LSTM** a batch size of 512 and a latent dimension of 128 were used.

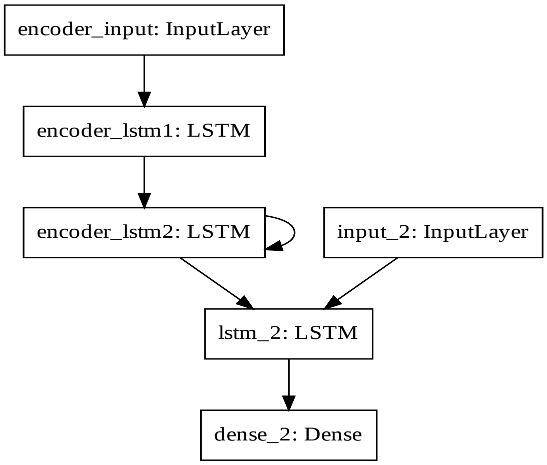

Fig. 4. **seq2seq_stacked_1_LSTM** model architecture

*3) Convolutional Sequence-to-Sequence (**ID: seq2seq_CNN**):* In our experiments, the CNN architecture contains 3 convolutional layers with 128 feature maps and a kernel of size 3, for both the encoder and the decoder networks. An attention architecture with a softmax activation follows the 3-layered convolutional decoder networks. The output is processed by another 2 convolutional layered architecture, with a softmax dense output. Figure 5 illustrates the model structure. The model is trained with a batch size of 1024 and a 128 latent dimension.

*4) *RNN and CNN hybrid model (**ID: seq2seq_hybrid**):* The RNN and CNN hybrid model [13] uses two paths - character level and word level. For the character path, an embedding layer feeds the input to 3 stacked CNN layers. The word path goes through embedding and a bidirectional LSTM (biLSTM). The two paths are merged by projecting words to characters based on a projection matrix which is received as an additional input. Hence, the character and word embeddings are jointly learned. These embeddings are fed to a stack of 3 convolutional layers. The output is predicted using a time distributed dense layer. We trained the network with a batch size of 32. The system architecture is illustrated in Figure 6.

*5) *RNN with language model:* In [5] a combination of character-level recurrent neural network based model and a language model are applied to automatic diacritics restoration.[3] The core model uses a bidirectional LSTM which deals with previous and next letter contexts in the sequence.

The bidirectional RNN contains 2 stacked layers with residual connections, composed of 300 LSTM units. A batch size of 200 was chosen. The model language is based on left-to-right beam search. At each time step, the output of the biLSTM layers is reduced by a fully connected layer to v-dimensional vectors, where v is the size of the output vocabulary. A non-linear ReLU activation function is applied to the reduced vectors. The final output layer uses a softmax activation.

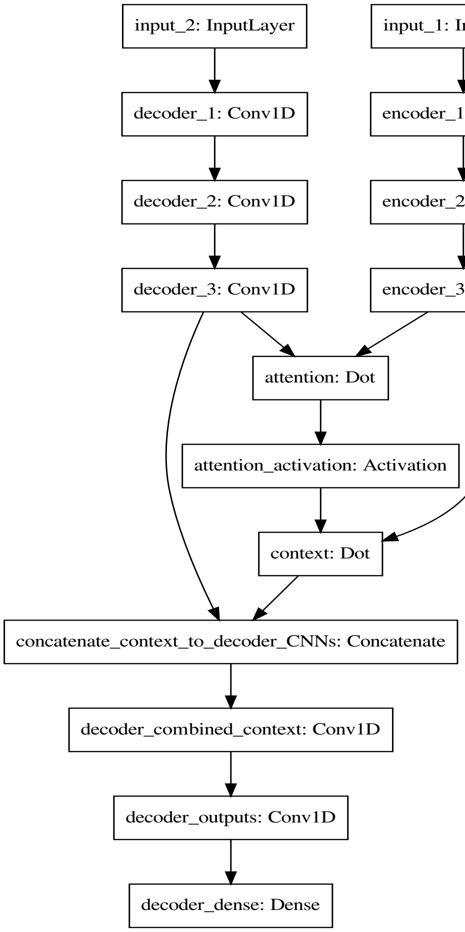

Fig. 5. **seq2seq-CNN** model architecture

V. Evaluation

All 6 system architectures were evaluated using the *classification accuracy metric*, which is defined as the ratio between the correct predictions and the total number of samples.

We computed the accuracy at three different levels: trigram, word and character level. At trigram and word-level the accuracy reflects the number of correct predictions made by the system overall. At character-level, we computed the accuracy only for the characters which may be written with diacritic symbols (*a, i, s, t*). Accuracy results for all the systems are presented in Table IV.

[3] https://github.com/arahusky/diacritics_restoration

TABLE IV
NETWORK PARAMETERS AND ACCURACY RESULTS

Architecture ID	Latent dimension	Batch size	Accuracy		
			3-gram level	Word level	Character level
seq2seq_LSTM	128	512	75.50%	89.98%	71.61%
seq2seq_stacked_1_LSTM	256	128	79%	93 %	78%
seq2seq_stacked_2_LSTM	128	512	84%	94 %	82%
seq2seq_CNN	128	1024	**91%**	**97 %**	89%
seq2seq_hybrid [13]	N/A	32	77%	92%	84%
seq2seq_LSTM_Language_model [5]	300	200	84 %	96 %	**90%**

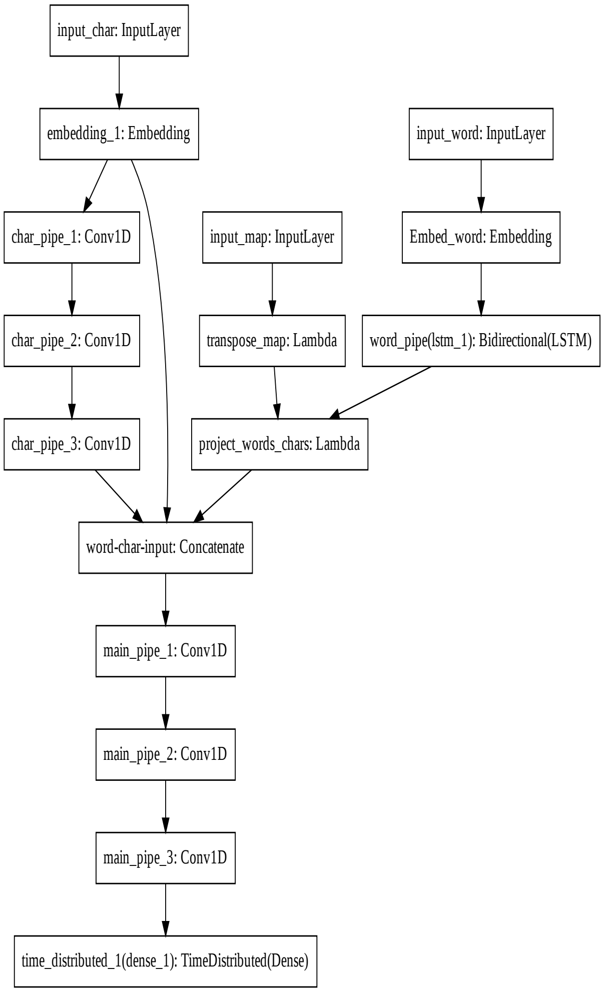

Fig. 6. **seq2seq-hybrid** model architecture

A separate set of results is shown in Table V, where the 4 ambiguous letter sets in Romanian (a-ă-â, i-î, s-ş, t-ţ) are analyzed individually.

TABLE V
ACCURACY RESULTS FOR INDIVIDUAL AMBIGUOUS PAIRS OF THE BEST PERFORMING SYSTEM

Architecture ID	Accuracy			
	a-ă-â	i-î	s-ş	t-ţ
seq2seq_CNN	93.51 %	99.44 %	98.39 %	97.94 %

The highest accuracy in terms of trigrams and words, was achieved for the convolutional network **seq2seq_CNN**, while the single-layer LSTM system **seq2seq_LSTM** had the lowest accuracy. One explanation can be found in the recurrence of the LSTMs, which may require larger data context, as opposed to the CNN, which uses an attention layer and sliding windows (kernels) to simulate the recurrence.

However, at character-level, the system described in [5] outperforms all other systems. The justification for this result can be the use in [5] of a language model together with the RNN, while our systems restore the diacritics without any additional linguistic information.

VI. CONCLUSIONS AND FUTURE WORK

In this work we compared 6 neural networks architectures for the task of automatic diacritics restoration applied to Romanian. All the models are trained using only parallel input-output pairs of texts with and without diacritics. As input to the sequence-to-sequence architectures we used character-level one-hot encodings. However, it is common practice in NLP to encode the words or characters using multidimensional embeddings obtained from large amounts of text data. These embeddings would allow the network to have an initial estimate of the characters' function in a language. So as future work, we intend to substitute the one-hot encoding with letter or word embeddings, and also to include additional linguistic or semantic information.

In our experiments we split the data in trigrams, both for training and for testing. Each network receives a diacritic-stripped trigram and predicts the entire corresponding sequence with diacritics. We intend to experiment with other N-gram, allowing the network to capture more context. One other means of improving the results is to predict the diacritics only for the sequence-ending word, considering all previous words to be correctly typed.

In addition, we are planning to investigate other types of fully convolutional neural networks, based on dilated convolutions combined with attention mechanisms, architectures largely used in Machine Translation and Speech Synthesis fields, but unexplored in the ADR domain.

Acknowledgment

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] D. Tufiş and A. Chiţu, "Automatic insertion of diacritics in romanian texts," in *Proceedings of the 5th International Workshop on Computational Lexicography COMPLEX*, 1999, pp. 185–194.

[2] M. Simard, "Automatic insertion of accents in french text," in *Proceedings of the Third Conference on Empirical Methods for Natural Language Processing*, 1998, pp. 27–35.

[3] R. Mihalcea and V. Nastase, "Letter level learning for language independent diacritics restoration," in *proceedings of the 6th conference on Natural language learning-Volume 20*. Association for Computational Linguistics, 2002, pp. 1–7.

[4] K.-H. Nguyen and C.-Y. Ock, "Diacritics restoration in vietnamese: letter based vs. syllable based model," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2010, pp. 631–636.

[5] J. Náplava, M. Straka, P. Straňák, and J. Hajic, "Diacritics restoration using neural networks," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.

[6] R. F. Mihalcea, "Diacritics restoration: Learning from letters versus learning from words," in *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 2002, pp. 339–348.

[7] C. Ungurean, D. Burileanu, V. Popescu, C. Negrescu, and A. Dervis, "Automatic diacritic restoration for a tts-based e-mail reader application," *UPB Scientific Bulletin, Series C*, vol. 70, no. 4, pp. 3–12, 2008.

[8] L. Petrică, H. Cucu, A. Buzo, and C. Burileanu, "A robust diacritics restoration system using unreliable raw text data," in *Spoken Language Technologies for Under-Resourced Languages*, 2014.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] H. Graves, Abdel-Rahman. Peephole long short-term memory, wikimedia commons. [Online]. Available: https://commons.wikimedia.org/wiki/

[12] V. B. Mititelu, E. Irimia, and D. Tufis, "Corola - the reference corpus of contemporary romanian language." in *LREC*, 2014, pp. 1235–1239.

[13] H. Cristescu. Romanian diacritic restoration with neural nets. [Online]. Available: https://github.com/horiacristescu/romanian-diacritic-restoration

Romanian Part of Speech Tagging using LSTM Networks

Beáta Lőrincz[a,b], Maria Nuţu[a,b], Adriana Stan[a]
[a]Communications Department, Technical University of Cluj-Napoca, Romania
[b]Department of Computer Science, "Babeş-Bolyai" University, Cluj-Napoca, Romania
{beata.lorincz, maria.nutu, adriana.stan}@com.utcluj.ro

***Abstract*—In this paper we present LSTM based neural network architectures for determining the part of speech (POS) tags for Romanian words. LSTM networks combined with fully-connected output layers are used for predicting the root POS, and sequence-to-sequence models composed of LSTM encoders and decoders are evaluated for predicting the extended MSD and CTAG tags. The highest accuracy achieved for the root POS is 99.18% and for the extended tags is 98.25%. This method proves to be efficient for the proposed task and has the advantage of being language independent, as no expert linguistic knowledge is used in the input features.**

***Index Terms*—POS tagging, recurrent neural networks, LSTM, sequence-to-sequence, Romanian, MSD, CTAG**

I. Introduction

Part of speech (POS) tagging is one of the key tasks of natural language processing (NLP). It refers to the identification of the part of speech of a given word and optionally, additional grammatical properties inherent to a particular POS. Along with other components of NLP, such as lemmatization, stemming, syllabification, word or sentence boundary detection and many others it aims to help computers understand and process natural language.

The difficulty of the task lays in the fact that the same orthographic form of a word can have a different meaning depending on the context (i.e. homographs). Another problem is that the declination and inflections of the words are not regular, especially in morphological rich languages, such as the case of Romanian. As a result, to define the correct POS of a word aside from its spelling, we also need to take into account the semantic links between words.

The task of POS tagging can have several annotation levels. The most basic one refers to determining the root POS (noun, verb, adjective, pronoun, determiner, article, adverb, adposition, conjunction, numeral, interjections, residual, abbreviation, part particle) and can usually be obtained from the dictionary entry of the word's lemma. The most complex tagset is the Morpho-Syntactic Descriptions (MSD) set, which adds several grammatical properties depending on the root POS [1]. Compared to MSD, the C-tagset introduced by [2] is a reduced size tagset that adds a maximum of 3 additional properties to the root POS. For ease of expression we will refer to the process of determining the root or an extended tag of a word as POS-tagging in general and denote the specific tagset where it is necessary.

Several single or multilingual text processing tools have been developed to perform the task of POS tagging with the scope of being incorporated into applications such as speech recognition, speech synthesis, machine translation or textual information extraction. The most common approaches among the previous studies rely on probabilistic and rule-based methods such as Hidden Markov Models, Maximum Entropy Classifiers, Bayesian Networks and Conditional Random Fields [3]. According to [3] these methods do not perform well on languages that use a lot of inflection, such as Romanian. In addition, rule based methods are language dependent and might require hand-crafted rules. Tools for building these rules have also been developed [4].

With the increasing popularity of machine learning, deep neural networks have also been successfully applied to NLP tasks. Neural network based algorithms have been used and compared to probabilistic POS-taggers since the 1990's [5] for the English language. To the best of our knowledge the first accuracy results for POS-tagging with neural networks applied to the Romanian language were reported in [3].

Studies concerning the Romanian language are also numerous with a relatively high reported accuracy. Tufiş et al. [2] achieved an accuracy of 98.39% using a tiered tagging with C-tagset for the language model, extended with a post-processor using probabilistic methods to reconstruct the MSD tag. [4] and [6] present hybrid methods combining statistical models with a rule based system that classifies tagging errors and reduces the need for manual rule construction. The best accuracy obtained by Simionescu is 97.03%. [3] reports an accuracy of 98.17% for MSD-tagging by implementing feed-forward neural networks with genetic algorithms used for designing the network topology. The BAILE multilanguage system [8] obtains an accuracy of 95.03% for POS-tagging for the Romanian language. Similar accuracy, 96.12% is achieved by [7] using a Naive Bayes model with a word database. All

TABLE I
POS-tagging accuracy results for Romanian reported in the literature

Authors	Method	Accuracy	Tagset
Tufis & Mason [2]	Probabilistic	98.39%	MSD
Boros & Dumitrescu [3]	Deep Neural Networks	98.19%	MSD
Simionescu [6]	Probabilistic & Rule-based	97.03%	MSD
Teodorescu et al. [7]	Probabilistic	96.12%	Root POS
Frunza et al. [8]	Machine Learning	95.30%	Root POS

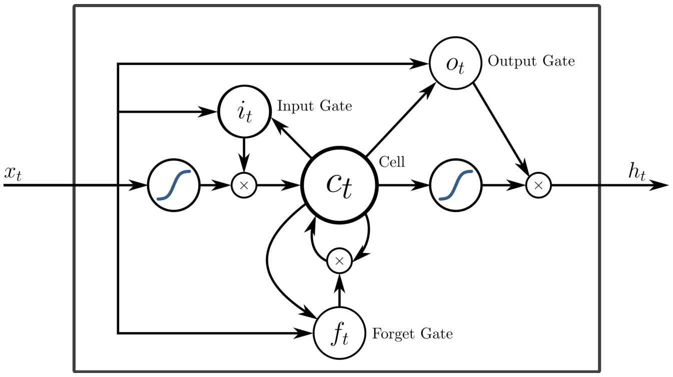

Fig. 1. LSTM memory cell [13]

the above results are summarized in Table I.

Starting from this overview, in this work we investigate the use of a recurrent neural network model with Long Short Term Memory (LSTM) layers for POS-tagging taking into consideration previous studies evaluated on several other languages (not including Romanian) [9], [10]. They conclude that LSTM networks work well for determining the POS-tag, not only with limited, but also with extended tagsets. We compare the results obtained with LSTM networks to a sequence-to-sequence model composed of LSTM encoders and decoders used for determining extended tags.

The paper is structured as follows: the proposed methods for POS-tagging are described in Section II, and the details of training data and implementation are elaborated in Section III. We discuss the results in Section IV. The conclusions and possible future work are summarized in Section V.

II. Method overview

As neural network based learning methods are now widely used in many areas of NLP and speech processing applications, in this article we focus on experimenting with different neural network architectures to determine the root POS or the extended tag.

Recurrent neural networks (RNN) are highly efficient in sequential data modelling. Their main advantage is that their output combines the current input with the output of the previous time step. Therefore it can extend its understanding of the data to the temporal connections between sequential inputs. However, vanilla RNNs cannot expand across long temporal sequences [11]. To overcome this disadvantage, Long Short Term Memory (LSTM) structures can be used [12]. These structures are based on recurrent nodes, but are extended with a memory cell which can model long-term dependencies in the input data. The memory cell contains a set of gates which decides what previous information needs to be saved in the cell state (memory), what needs to be discarded and how much the current input and stored memory will influence the output of the cell. The memory cell components are depicted in Figure 1.

The behaviour of the cell gates are described by Equations 1, 2, and 3:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{1}$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{2}$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{3}$$

where σ is the sigmoid function, and i, f, o are the input, forget, output gates, respectively, at time step t. w_x are the weights of the appropriate *gate(x)*, x_t is the input at time step t and b_x is the bias for the respective *gate(x)*. The states of the cell at the current time step t are stored in cell vector c_t and hidden vector h_t. These state vectors have the same dimension as the cell gate vectors. Equations 4 and 5 describe the state vectors:

$$c_t = f_tC_{t-1} + i_t \tanh\left(w_c[h_{t-1}, x_t] + b_c\right) \tag{4}$$

$$h_t = o_t \tanh c_t \tag{5}$$

where c_t – the cell state is computed based on the previous state with a sigmoid activation function computed in training time and with the candidate for the current state.

The hidden state is calculated from the cell state passed through an activation function and element-wise multiplied with the output vector at time step t.

In our experiments to predict the root POS-tag (containing a single character) we added to the LSTM layers a fully-connected (dense) layer. The unit size of this layer is set to the number of possible POS-tags. The behaviour of this layer is described in Equation 6:

$$o_x = (Wx) + b \tag{6}$$

where the output is calculated based on the inputs (x), weight (W) and bias (b).

To determine the MSD tag of the word, a sequence-to-sequence (seq2seq) [14] learning method was applied. RNN networks can be used when the input and output vectors can be encoded with fixed-dimension vectors and are not appropriate for defining the variable length MSD tag of a word. The seq2seq model is composed of an *encoder* and a *decoder*. The encoder and decoder are both neural networks that are frequently implemented with LSTM cells.

The encoder is responsible for interpreting the input data, one time step at a time, and converting it into a fixed dimension vector representation. The decoder uses the hidden or output state of the encoder to condition its own output. In general, the decoder is trained with one time step-delayed sequences. This means that it learns to predict the next character or word in the output sequence. Figure 2 presents the seq2seq model for the word ”acasă” as input and ”Rg” as output sequence, where <SS> marks the start and <SE> the end of sequence.

III. Evaluation

A. Tagsets

Different tagsets can be used for POS-tagging to indicate the POS and/or additional grammatical categories. The tagset is language dependent and contains all possible parts of speech in the respective language. For Romanian, we used the following tagsets: root tagset (ID:**RPOS**), MSD-tagset (ID:**MSD**) and

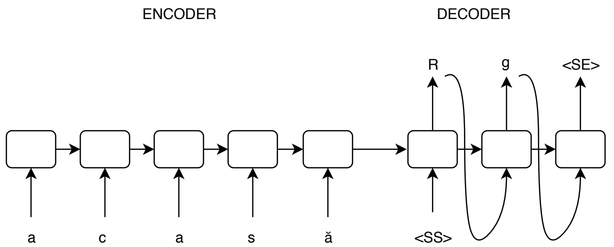

Fig. 2. Sequence-to-sequence model input and output sequence example

TABLE II
NUMBER OF TAGS PER TAGSET

Tagset	No. of tags used
Basic	13
MSD	334
CTAG	89

TABLE III
MSD TAG EXAMPLES FOR A VERB AND A NOUN

Vmp3pf		Ncmsrn	
V	verb	N	noun
m	main	c	common
p	participle	m	masculin
3	third person	s	singular
p	plural	r	nominative
f	feminine	n	not definite

C-tagset (ID:**CTAG**). The number of different tags used from each tagset is shown in Table II.

The most basic tagset contains only the part of speech of the word's lemma as extracted from a linguistic dictionary (e.g 'V' for verb, 'N' for noun, 'A' for adjective etc.). We refer to this as the *root POS* tag.

The MSD (Morpho-Syntactic Descriptions) is an extended tagset containing the codes defined by the MULTEXT-EAST project [1]. The number of MSD tags varies per dataset and according to [15] a number of 615 tags were described for the Romanian word-form lexicon. The MSD is a hierarchical POS representation, where the first upper case letter represents the root POS followed by the lexical attributes of the POS. Examples for a verb and noun MSD tag are shown in Table III.

A different tagset, called C-tagset (CTAG) was introduced in [2]. CTAG is a reduced version of the MSD and can be mapped to it directly. For example 'NSRN' stands for a common noun, that is singular, direct and indefinite. We used this tagset when training on large amounts of text where the tag is defined based on the context of the word.

B. Datasets

Three different datasets were used for the training part. Most of the experiments were run on the morphological dictionary created by Simionescu [4] with the help of DexOnline database and Wikipedia[1] proper nouns collection (ID:**WPT**).[2] This dataset consists of 1,715,881 words associated with one or more MSD tags. For the POS-tagging training we used the first character of the MSD tag. If multiple tags were available for the same orthographic form, only the first entry was used. For evaluation purposes the test data was validated against all possible tags assigned to the word sample.

TABLE IV
NUMBER OF TRAINING AND TEST SAMPLES PER DATASET

Dataset	Total samples	No. of training samples	No. of test samples
WPT	1,715,881	897,328	224,331
DEX	1,994,412	936,611	234,152
CoRoLa	3,075,165	2,460,132	615,033

The second dataset we used is the Romanian Explicative Dictionary database (ID:**DEX**)[3] that contains 1,994,412 words, each associated with one or more POS-tags and a word frequency. The words with 0 frequency were removed from the training, resulting in a set of 1,158,197 samples.

The third dataset consists of 125,316 sentences selected from the CoRoLa [16] corpus (ID:**CoRoLa**).[4] The sentences contain 3,075,165 words with a number of 175,946 unique words. The CoRoLa corpus provides linguistic attributes for all words including the MSD and CTAG annotations and contains texts of different styles such as juridical or scientific. The word samples were collected from these texts randomly.

The first two datasets contain individual words with single or multiple POS or MSD information attached. The third dataset provides context-related information as the tags are assigned to words of the sentences. From each dataset 20% of the samples was randomly selected for testing and the remaining 80% used for training. The number of train and test samples is summarized in Table IV.

C. Data input format

The input data was coded either with one-hot-encoding (ID:**OHE**) or letter embedding (ID:**LE**). In the former encoding the words are represented by a two-dimensional binary matrix. The size of the matrix depends on the number of input characters and the maximum length of the input words. The letter embedding uses a dense representation of each character that contains information about the context of the selected character which is important for POS-tagging [17]. The Gensim[5] library was used to create a letter embedding of order 30 based on the Romanian Wikipedia pages' database dump. The order was selected to be close to the number of characters in the Romanian alphabet.

Word embeddings were not considered in our study, as the focus was to predict the POS tags using only the orthographic form of the words, independent of their linguistic context.

[1] https://ro.wikipedia.org/
[2] http://nlptools.infoiasi.ro/WebPosTagger
[3] https://dexonline.ro/
[4] http://corola.racai.ro/
[5] https://radimrehurek.com/gensim/

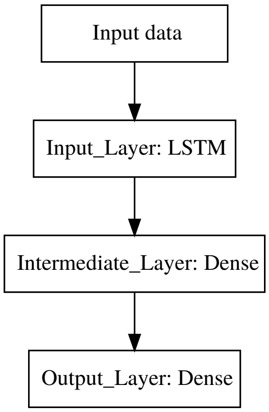

Fig. 3. LSTM with dense layers model

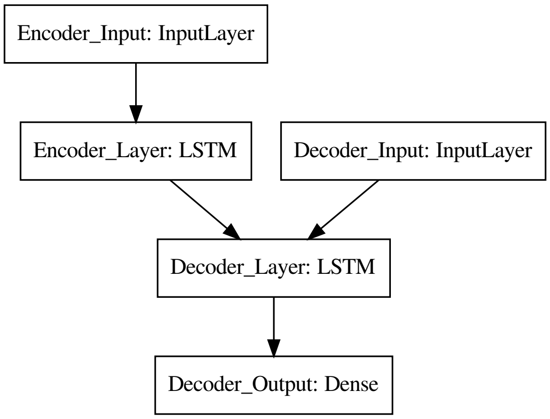

Fig. 4. LSTM sequence-to-sequence model

D. System architectures

For the implementation we used the Keras[6] deep learning library with TensorFlow[7] backend. We grouped our network architectures into the following categories:

1) LSTM with fully-connected (dense) layers: The input data was one-hot encoded or used letter embeddings and provided to a first LSTM layer. The latent dimension of the LSTM layer varied from 64 to 1024. On top of the LSTM layer two dense layers were stacked. The second dense layer serves as the output layer, having the size equal to the number of possible POS-tags. The architecture of this network is shown in Figure 3.

2) Sequence-to-Sequence Model: A character-level sequence-to-sequence (seq2seq) model is used for the MSD tagging. The input sequences are the words and the MSD tags of the target sequences. The encoder LSTM layer converts the input sequence into state vectors. The decoder LSTM layer uses the initial state vectors of the encoder and turns the target sequences into the same sequence offset by one time step. Teacher enforcing is used to generate the next character, as the decoder generates the $target[t+1,...]$ sequence given the sequence $target[...,t]$. We add a starting and ending character to each target sequence. An example of an input and target sequence composed of 4 words is shown below:

Input sequence: 'Absolvent al Facultăţii de'
Target sequence: '#NSN TS NSOY S@'

When an unknown input sequence is decoded, we begin with the starting character and use the decoder to predict the next character until the end of sequence character or a maximum sequence length is reached.

The sequence-to-sequence model was used to determine the CTAGs within the context of sentences. The encoder and decoder models were trained with a sliding window over the words. The networks we trained used 3 to 5 words and were shifted to the right by one word for each sample. The best accuracy for the tags was achieved when the training samples were composed of 3 words. The architecture of this network is shown in Figure 4.

IV. Results and discussions

Each network was evaluated on 20% of the initial dataset held out of the training data. The efficiency of the training method was measured by accuracy: dividing of the number of correct predictions by the total number of predictions. When calculating the accuracy for the MSD tags we only considered as correct outputs the predicted sequences that fully matched the target MSD-tag. The parameters and accuracy results are shown in Table V. The asterisk (*) marks the systems where we considered all possible tags associated with the words as correct answers, and not only the first entry tag.

The test data was randomly selected, and both training and test samples were shuffled before training. Based on initial tests we choose to set the batch sizes to 256, 512 or 1024. The best results were obtained with a batch size of 512. The latent dimension of the LSTM cells was evaluated with 64, 128, 256, 512 respectively 1024, the best accuracy numbers were achieved with size 256.

The best accuracy for the root POS-tagging (when predicting only a single character) was 99.18% (System ID 1). This outperforms all the systems presented in Table I which predict the root POS. When the same network was validated against only the most frequent POS and not all the possible POS tags available in the dataset, the accuracy was 94.85% (System ID 2). The accuracy breakdown per root POS is shown in Table VI. The POS types that did not end up in the test set due to their low count and the random selection, are marked with a not applicable (N/A) accuracy. The loss was calculated

[6] https://keras.io/

[7] https://www.tensorflow.org/

TABLE V
NETWORK PARAMETERS AND ACCURACY RESULTS

System ID	Dataset	Tag	Network type	Character encoding	Latent dimension	Batch size	Epochs	Accuracy
1	WPT	RPOS	LSTM + Dense (*)	OHE	256	512	50	**99.18%**
2	WPT	RPOS	LSTM + Dense	OHE	256	512	50	94.85%
3	WPT	RPOS	LSTM + Dense	LE	256	256	25	54.80%
4	WPT	RPOS	seq2seq LSTM	LE	256	256	25	94.99%
5	WPT	RPOS	seq2seq + Embedding layer	OHE	256	256	20	93.88%
6	WPT	MSD	seq2seq LSTM (*)	OHE	512	1024	50	98.25%
7	WPT	MSD	seq2seq LSTM	OHE	512	1024	50	75.28%
8	WPT	MSD	seq2seq + Embedding layer	OHE	256	512	50	76.62%
9	DEX	RPOS	LSTM + Dense	OHE	256	512	50	94%
10	CoRoLa	CTAG	seq2seq LSTM	OHE	256	512	100	97.15%

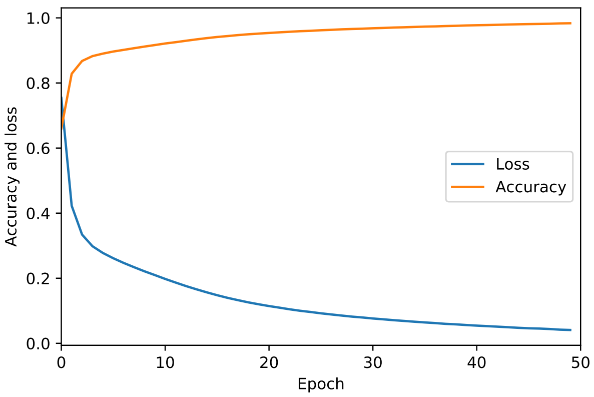

Fig. 5. The model accuracy and loss during training for System ID 1

TABLE VI
ACCURACY SUMMARY PER POS FOR SYSTEM ID 1

POS	No. of train samples	No. of test samples	Accuracy
Noun	471470	118133	99.86%
Adjective	232731	58457	99.86%
Verb	191489	47320	99.76%
Adverb	783	212	98.29%
Numeral	214	65	98.21%
Pronoun	220	48	97.76%
Determiner	161	37	98.99%
Interjections	156	37	96.37%
Abbreviation	58	0	N/A
Adposition	39	9	97.92%
Conjunction	12	0	N/A
Article	6	0	N/A
Part Particle	2	0	N/A

with categorical cross entropy method. The training accuracy and loss values of this model are shown in Figure 5.

The sequence-to-sequence model achieved an accuracy of 98.25% for the MSD tags (System ID 6). Compared to the POS prediction where the model only needed to predict one of the 13 root POS tags, the MSD model needs to predict 334 possible tags. Our results are comparable to the ones reported in [2]. However, the evaluation datasets are different, and [2] also uses context information.

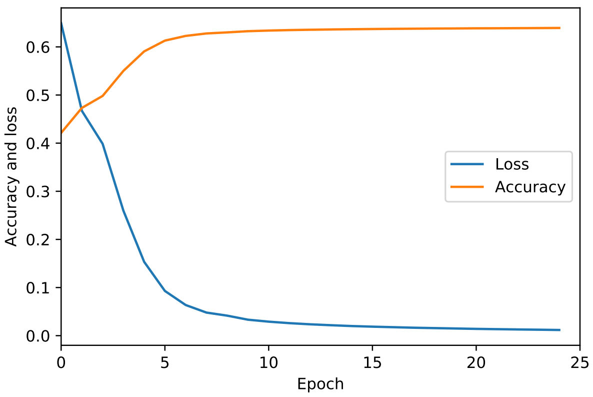

Fig. 6. The model accuracy and loss during training for System ID 6

The sequence-to-sequence model on CTAGs obtained an accuracy of 97.15%. The loss and accuracy values during model training are shown on Figure 6. This model uses context related information which improves the performance and could be extended to hierarchically predict MSD tags.

The models were tested with letter embeddings as input and also with the input layer replaced by an embedding layer. Changing the input did not result in higher accuracy. This could be explained by the fact that in the case of LE the positional information of a letter does not change the POS. In the case of the embedding layer the available data might not be sufficient for a good representation learning of the input.

V. CONCLUSIONS AND FUTURE WORK

In this paper we evaluated two different types of neural network architectures for POS-tagging, applied on the Romanian language. LSTM networks with fully-connected layers performed well on predicting the root POS, while sequence-to-sequence models achieved high accuracy for defining extended tags, such as the MSD and CTAGs. The addition of the letter embeddings and using embedding layers instead of the one-hot-encoding for representing the input data did not result in higher accuracy.

Despite the very good results obtained by our network architectures, there is still a need to explore other network architectures, such as convolutional neural networks, especially for the extended tagsets. Adding other linguistic information, such as lemma or lexical stress could improve the accuracy of the POS taggers, especially in the case of homographs.

Acknowledgment

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] T. Erjavec, "Multext-east morphosyntactic specifications: Version 3.0," *Supported By EU Projects Multext-East, Concede And TELRI*, 2004.

[2] D. Tufis and O. Mason, "Tagging romanian texts: a case study for qtag, a language independent probabilistic tagger," in *Proceedings of the First International Conference on Language Resources and Evaluation (LREC)*, vol. 1, no. 589-596, 1998, p. 143.

[3] T. Boroş and S. D. Dumitrescu, "Improving the racai neural network msd tagger," in *International Conference on Engineering Applications of Neural Networks*. Springer, 2013, pp. 42–51.

[4] R. Simionescu, "Graphical grammar studio as a constraint grammar solution for part of speech tagging," in *The Conference on Linguistic Resources and Instruments for Romanian Language Processing*, vol. 152, 2011.

[5] H. Schmid, "Part-of-speech tagging with neural networks," in *Proceedings of the 15th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 1994, pp. 172–176.

[6] R. Simionescu, "Hybrid pos tagger," in *Proceedings of Language Resources and Tools with Industrial Applications Workshop (Eurolan 2011 Summer School), Cluj-Napoca, Romania*. Citeseer, 2011, pp. 21–28.

[7] L. R. Teodorescu, R. Boldizsar, M. Ordean, M. Duma, L. Detesan, and M. Ordean, "Part of speech tagging for romanian text-to-speech system," in *2011 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. IEEE, 2011, pp. 153–159.

[8] O. Frunza, D. Inkpen, and D. Nadeau, "A text processing tool for the romanian language," in *Proc. of the EuroLAN 2005 Workshop on Cross-Language Knowledge Induction*. Citeseer, 2005.

[9] T. Horsmann and T. Zesch, "Do lstms really work so well for pos tagging?–a replication study," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 727–736.

[10] B. Plank, A. Søgaard, and Y. Goldberg, "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss," *arXiv preprint arXiv:1604.05529*, 2016.

[11] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[13] H. Graves, Abdel-Rahman. Peephole long short-term memory, wikimedia commons. [Online]. Available: https://commons.wikimedia.org/wiki/

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[15] D. Tufiş, "Tiered tagging and combined language models classifiers," in *International Workshop on Text, Speech and Dialogue*. Springer, 1999, pp. 28–33.

[16] V. B. Mititelu, E. Irimia, and D. Tufis, "Corola—the reference corpus of contemporary romanian language." in *LREC*, 2014, pp. 1235–1239.

[17] C. D. Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1818–1826.

Input Encoding for Sequence-to-Sequence Learning of Romanian Grapheme-to-Phoneme Conversion

Adriana STAN
Communications Department
Technical University of Cluj-Napoca, Romania
adriana.stan@com.utcluj.ro

***Abstract*—This paper evaluates the use of sequence-to-sequence learning models for the Romanian grapheme-to-phoneme conversion. The strategies explore the use of different input feature encoding: one-hot letter encoding, additional embedding layer and grapheme embeddings learned from a large corpus of Romanian text. Additional lexical information, such as syllabification and lexical stress is also taken into consideration for augmenting the orthographic form of the word and providing more accurate phonetic transcriptions.**

The sequence-to-sequence models are also compared to a baseline decision tree algorithm in terms of both phone- and word-level accuracy. The best results are achieved by the model which uses grapheme embeddings and all additional linguistic information. Its accuracy is 97.90% at word-level, and 99.62% at phone-level. However, only minor differences exist between the tested systems.

***Index Terms*—phonetic transcription, Romanian, grapheme-to-phoneme, G2P, LTS, LSTM, DNN, letter embeddings, grapheme embeddings**

I. INTRODUCTION

The latest trends in human-computer interaction using natural language or speech processing tend to limit the input feature manipulation by the human expert in favor of a deep learning architecture. This architecture can, at least in theory, extract similar features on its own. For example, state-of-the-art speech synthesis systems are now trained using only pairs of text and audio with no segment alignments or text processing [1], [2]. Machine translation systems can learn to map sequences of texts in two separate languages, without the need of an annotated correspondence between the words of the sequences.

Even though the additional features are not required in these particular tasks, they are still an essential part of many other applications. For example, the lexical stress or phonetic transcription can discriminate between homographs in semantic analysis. To develop algorithms which can extract these features with high accuracy, large lexicons and datasets are required. In this respect, Romanian is an under resourced language [3] and does not benefit from readily available, high quality language resources and systems. However, in recent years more and more resources and tools have been developed and published by the research community.

One of the most important linguistic resources for Romanian was developed within the CoRoLa project [4], [5]. The CoRoLa corpus of Romanian texts contains over 1 billion tokens. The data is cleaned, partially annotated with expert linguistic metadata and includes multiple text styles. The datasets can be accessed through an online application.[1]

The open-source initiative of the online Romanian Explicative Dictionary [6] is also a great resource for extracting word inflections or part-of-speech tags, as well as the syllabic forms of the words which do not follow the general syllabification rules in Romanian.[2]

Scientific publications of Romanian textual resources include: Barbu et al. [7] published two Romanian dictionaries of syllabic and inflected forms for over 500,000 Romanian words of 65,000 different lemmas. Simionescu released an extensive list of part-of-speech annotations for over 1,000,000 words [8].[3] Domokos et al. [9] developed an extensive grapheme-to-phoneme dictionary in SAMPA format, called NAVIRO, and built from an initial list of 10,000 manually transcribed words, and extended using artificial neural networks.[4] Another freely available grapheme-to-phoneme dictionary containing over 75,000 entries composed mainly of the Romanian Scrabble's Association's official list of words was introduced in [10].[5]

With the availability of these resources, other studies and tools have been published, investigating either entire text processing systems or only parts of them. The following listing includes only those works related to the task of grapheme-to-phoneme conversion or complete systems which contain the phonetic conversion among other processes.

In [11] the use of neural networks for grapheme-to-phoneme conversion in the context of text-to-speech synthesis is investigated, and a 98.3% word-level accuracy is reported. A similar approach, based on artificial neural networks, is presented in [12] and [13]. The authors of [13] report a 92.83% accuracy at phone-level. A set of rules for the Romanian grapheme-to-phoneme transcription was developed in [14] and obtained 95% accuracy. Similar rule-based transcriptions combined with decision trees are explored in [15]. The accuracy of the method at word-level is 94.8% on one of the evaluation subsets. The authors of [16] compare 5 separate methods based on: decision trees, neural networks, support vector machines,

[1] http://corola.racai.ro/
[2] http://dexonline.ro
[3] http://nlptools.infoiasi.ro/WebPosTagger
[4] http://users.utcluj.ro/~jdomokos/naviro/
[5] http://speech.utcluj.ro/marephor/

pronunciation by analogy and an expert system. Their best results have an accuracy of 96.68% at word-level.

The work in [17] achieves a 93% word-level accuracy for the Romanian grapheme-to-phoneme conversion using a maximum entropy classifier and a custom data-driven algorithm. [18] introduces a margin infused relaxed method and a specialized algorithm for same processing task. The reported word-level accuracy is 96.29%. The same authors present in [19] a series of decision tree-based evaluations for multiple Romanian and English text processing tasks, including grapheme-to-phoneme conversion. The word-level accuracy of the G2P module was reported at 95.05%.

A novel approach to the phonetic transcription task was presented in [20] where grapheme-to-phoneme conversion is performed using statistical machine translation principles and obtains a 97.24% accuracy at word-level. The authors of [10] use decision trees with various context lengths and achieve a 99.61% accuracy at phone-level. The decision trees are compared with deep learning networks in [21]. The best algorithm obtained at most 99.63% accuracy at phone-level.

Table I summarizes these works and their reported accuracy. However, the methods are not directly comparable as the training datasets are not consistent across the studies. Also, the level of the reported accuracy (i.e. phone- or word-level) differs.

It is also worth mentioning a few papers which introduced full text processing systems, such as [22] which addresses the diacritic restoration, text normalization, syllabification, phonetic transcription and lexical stress positioning; [23] describes a complete text-to-speech synthesis system including the front-end text processing with syllabification, lexical stress assignment and grapheme-to-phoneme conversion. [24] presents the authors' work on a full text processing tool for phonetic transcription, syllabification and part-of-speech tagging.

In terms of grapheme-to-phoneme conversion for other languages, the most recent approaches make use of the complex deep learning architectures. Studies on this topic are numerous and include various mono- or multi-lingual tasks, as well as the use of unsupervised representation learning for the graphemes: [25] introduces a bidirectional long short-term memory architecture combined with alignment-based models for translating English words into their phonetic representations. [26] adapts good G2P models for low-resource languages in a multilingual framework. [27] experiments with uni- and bi-directional LSTMs with various output delays combined with an n-gram language model. [28] uses a multitask learning strategy combined with n-gram language models to improve the G2P of English and German texts. [29] learns global character vectors from plain text resources and applies them to monolingual and multilingual G2P conversion in a recurrent neural network setup.

TABLE I
RESULTS REPORTED IN THE LITERATURE FOR THE TASK OF GRAPHEME-TO-PHONEME CONVERSION IN ROMANIAN

Paper	**Reference**	**Level**	**Accuracy**
(Burileanu, 2002)	[11]	word	98.30%
(Ordean et al., 2009)	[15]	word	94.80%
(Toma et al., 2009)	[14]	word	95.00%
(Domokos et al., 2011)	[13]	phone	92.83%
(Toma et al., 2013)	[16]	word	96.68%
(Boroş et al.,2012)	[17]	word	93.00%
(Boroş et al., 2013)	[18]	word	96.29%
(Cucu et al., 2014)	[20]	word	97.24%
(Boroş et al. , 2017)	[19]	word	95.05%
(Toma et al., 2017)	[10]	phone	99.61%
(Stan et. al, 2018)	[21]	phone	99.63%

Starting from this overview, the aim of this paper is to investigate the use of the newly developed sequence-to-sequence deep learning models [30] applied to the Romanian grapheme-to-phoneme conversion. The method evaluates a simple encoder-decoder structure with different grapheme input encoding and compares the results with those of a basic decision tree algorithm.

The paper is organized as follows: Section II describes the sequence-to-sequence algorithm and its internal network structure. Section III introduces the datasets and evaluation procedures, while conclusions and discussions are presented in Section IV.

II. Sequence-to-Sequence Learning for Grapheme-to-Phoneme Conversion

The task of grapheme-to-phoneme conversion implies variable length input and output sequences, i.e. the length of the word versus its phonetic transcription. The recently introduced sequence-to-sequence models [30] can do just that. Their structure involves an *encoder-decoder* architecture. The encoder aims to create a low-dimensional representation of the input sequence which captures the essential information for the task at hand. The output of the encoder is then used to condition the decoder's output. The decoder is trained on time-delayed sequences, meaning that it learns to predict the next token of the sequence. Both the encoder and decoder are neural networks with recurrent or convolutional structures [31].

This work uses the recurrent architecture. Recurrent neural networks (RNN) are a class of neural networks in which connections between the nodes are made so that temporal sequences can be accurately modeled [32]. This means that the output at time step t_i is conditioned on the state of the network at time step t_{i-1} along with the current input.

However, in vanilla RNNs, sequences with long temporal dependencies cannot be represented properly. The solution to this problem is to use a specialized type of neural node able to model longer temporal dependencies in the input data. One such node is the Long Short Term Memory (LSTM) [33]. LSTMs have a more complex internal structure (see Figure 1) which includes a series of gates designed to either allow, partially allow or block the current information to pass to the next time step. LSTMs have been successfully applied to several complex tasks in NLP, such as machine translation [30], language modeling [34] or text generation [35].

The equations describing the behavior of the LSTM are:

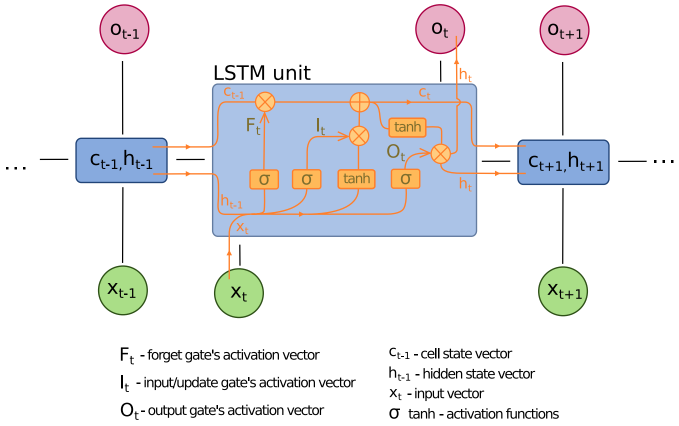
o_{t-1}
o_t
o_{t+1}
LSTM unit
h_t
c_{t-1}
c_t
F_t
I_t
tanh
O_t
c_{t-1}, h_{t-1}
c_{t+1}, h_{t+1}
σ
σ
tanh
σ
h_t
h_{t-1}
x_t
x_{t-1}
x_t
x_{t+1}
F_t - forget gate's activation vector
I_t - input/update gate's activation vector
O_t - output gate's activation vector
c_{t-1} - cell state vector
h_{t-1} - hidden state vector
x_t - input vector
σ tanh - activation functions

Fig. 1. Diagram for a one-unit Long Short-Term Memory (LSTM) [36]

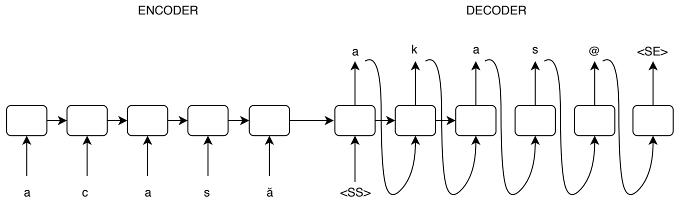

Fig. 2. Sequence-to-sequence model architecture. <SS> and <SE> mark the sequence-start and sequence-end tokens, respectively.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{1}$$

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3}$$

$$c_t = f_t C_{t-1} + i_t \tanh\left(W_c[h_{t-1}, x_t] + b_c\right) \tag{4}$$

$$h_t = o_t \tanh(c_t) \tag{5}$$

where σ and $tanh$ are the sigmoid and hyperbolic tangent functions, respectively. x_t is the input vector to the LSTM unit, i_t is the input/update gate's activation vector, f_t is the forget gate's activation vector, and o_t is the output gate's activation vector at time step t. W and b are the corresponding weight matrices and bias vectors for each gate–these parameters are learned in the training process. c_t is the cell state vector at time step t, and h_t is the hidden state vector of the LSTM unit. h_t is also known as the output vector of this cell type.

Across all experiments the same sequence-to-sequence model architecture was used. The architecture is composed of a single layer LSTM in both the encoder and the decoder, as shown in Figure 3. However, several input sequence encoding strategies were evaluated: *one-hot encoding*, *embedding layer* and *grapheme embeddings*. All encoding was performed at *grapheme-level*.

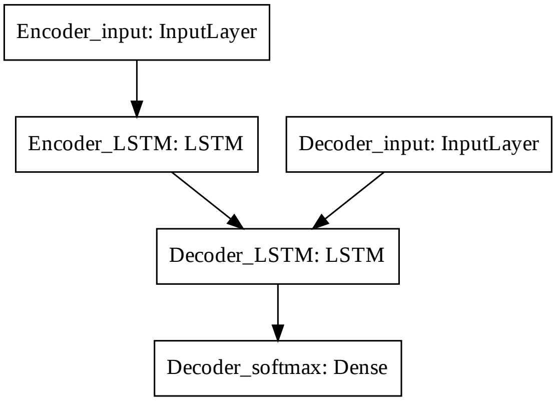

Fig. 3. Sequence-to-sequence network architecture

The **one-hot encoding (OHE)** simply creates a 2D array with the dimension MxN, where M is equal to the maximum length of the input sequences, and N is equal to the number of distinct characters available in the input sequences. Each row has null elements except for one which is equal to 1. The index i of the element equal to 1 is determined as the position of the current input character c in the set of all possible input characters $\{c_1, c_2, ..., c_i, ..., c_N\}$. For example, if the set of all possible input characters is $\{a, b, c\}$ and we want to encode the sequence $aababc$ the OHE of this sequence is:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The **embedding layer (EL)** is an additional neural layer added between the encoder's input and the LSTM. The embedding layer is jointly learned with the rest of the sequence-to-sequence model's parameters. The task of the embedding layer is to convert the input characters in each sequence into dense vectors of fixed-length. The dense vectors should provide better approximations of the input characters' relevance to the given learning task.

However, the training data in most NLP applications is rather limited. Therefore, a fixed-length embedding of the input characters could be learned from an external text resource, such as large amounts of raw texts. This embedding would approximate the context in which each character from the set can be found in a particular language. This **grapheme embedding (GE)** is similar to the word embeddings [37]. The difference is that vectors are obtained at character/letter level. Word embeddings have proved their efficiency in a multitude of NLP tasks [38], and grapheme embeddings were shown to mimic the behavior of the phonetic transcription [39].

TABLE II

Example of input-output sequences for the word *acasă* (en. *home*) using the orthographic form, orthographic plus syllabification, orthographic plus lexical stress, and all 3 combined.

Type	**Input seq**	**Output seq**
Orthographic	acasă	a k a s @
Orthographic+Syllabification	a-ca-să	a k a s @
Orthographic+Lexical stress	ac'asă	a k a s @
Orthographic+Syllabification+Lexical stress	a-c'a-să	a k a s @

III. Evaluation

A. Training data

To evaluate the performance of sequence-to-sequence models applied to Romanian grapheme-to-phoneme conversion, the MaRePhor [10] phonetic dictionary was used.[6] The dictionary consists of 72,375 words and 591,570 letters. The entries are words from the Romanian Scrabble Association's official list of words and the entries from a 15,517 words dictionary developed according to the SpeechDat specifications. The phonetic transcriptions are in SAMPA format.[7]

Aside from the direct grapheme-to-phoneme conversion, additional linguistic information was added to the input data with the aim of aiding the phonetic transcription. Syllabification and lexical stress of the words were appended to the input features. The syllabification of approximately 507,000 words was extracted from the RoSyllabiDict lexicon [7]. The DEX Online Database [6] which includes over 1,600,000 words and their inflected forms along with the stress labels was selected for lexical stress assignment.

Combining the common words in all 3 dictionaries a list of 62,874 words was obtained. The syllabification and lexical stress were incorporated into the orthographic form of the word, either individually or simultaneously. The output sequence was in all experiments only the phonetic transcription. Table II shows an example of the entries. The data was randomly split into training (80%) and testing (20%) sets. The split was maintained across all evaluations so that there are no differences between the systems caused by the random split of the training and testing datasets.

The Wikipedia database dump for Romanian [40] was processed through a word2vec model using the Gensim toolkit[8] to obtain the grapheme embeddings. The order of the embedding was to set to 30 so that it is close to the number of letters in the Romanian alphabet (i.e. 31 letters).

A plot of the t-SNE [41] visualizations of the grapheme embeddings is presented in Figure 4. It can be noticed that the vowels *a, e, i, o, u* are closely grouped together. The same grouping can be found for the least frequent letters in Romanian *k, w, y, q*. This means that, at least in theory, the network can make use of better representations for its inputs.

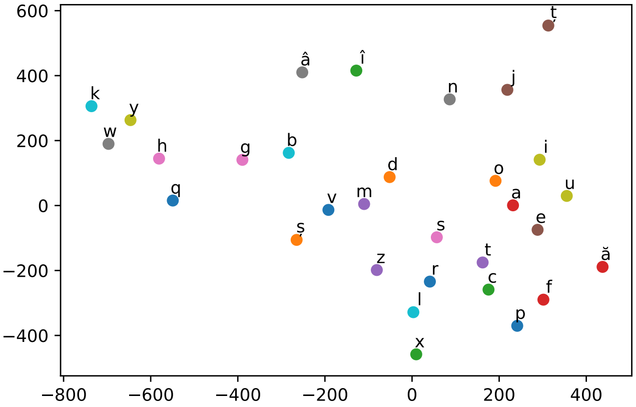

Fig. 4. Bidimensional t-SNE plot of the Wikipedia-based Romanian grapheme embeddings

B. Results

The neural sequence-to-sequence architecture for the Romanian grapheme-to-phoneme task, was implemented using the Keras[9] toolkit with TensorFlow backend.[10] Because of the non-sequential structure of the network, the functional API of Keras was used. This enables us to merge the input of the decoder with the hidden state of the encoder.

As a first step in the evaluation, we explored the hyper-parameter setup of the network. Initial tests were carried out for the dimension of the batch size and the latent dimension of the LSTM layers in the encoder and decoder. The best results were obtained using a batch size of 32 and a latent dimension of 512 nodes. The embedding layer consists of 30 nodes so that there is a correspondence between the externally-learned grapheme embeddings and this one. The weights of the network were optimized using RMSprop, an algorithm similar to Adagrad [42] which tunes the learning rate depending on an running average of the recent gradients. The loss function was set to categorical cross-entropy due to the multiclass output of the decoder. The number of epochs was set to 20 for the OHE and GE encodings. For the EL encoding, because the network has to learn additional weights, the number of epochs was set to 40.[11].

Accuracy results of the 3 sequence-to-sequence networks with their respective input embeddings are presented in Tables III, IV and V. The accuracy is measured at phone- and word-level. Phone-level accuracy took into account all the predicted phones, and not just the ones that pose problems in Romanian (see [10]). It can be noticed that the GE input achieves the highest accuracy scores: **99.51%** at phone-level and **97.40%** at word-level when considering only the orthographic form of the word as input. If the syllabification and

[6] https://speech.utcluj.ro/marephor/

[7] https://www.phon.ucl.ac.uk/home/sampa/romanian.htm

[8] https://radimrehurek.com/gensim/

[9] https://keras.io/

[10] https://www.tensorflow.org/

[11] Jupyter notebooks for all systems' training flow are available here: http://github.com/speech-utcluj/

TABLE III
ACCURACY RESULTS FOR **SEQ2SEQ** WITH **ONE-HOT ENCODING**

	Accuracy [%]	
Input features	**Phone**	**Word**
Orthographic	99.42	97.05
Orthographic+Syllabification	99.12	96.40
Orthographic+Lexical stress	99.01	95.85
Orthographic+Syllabification+Lexical stress	99.40	97.45

TABLE IV
ACCURACY RESULTS FOR **SEQ2SEQ** WITH **WIKI-BASED GRAPHEME EMBEDDINGS**

	Accuracy [%]	
Input features	**Phone**	**Word**
Orthographic	99.51	97.40
Orthographic+Syllabification	99.30	96.40
Orthographic+Lexical stress	99.56	97.70
Orthographic+Syllabification+Lexical stress	**99.62**	**97.90**

the lexical stress are added to the input features, the accuracy increases to **99.62%** at phone-level and to **97.90%** at word level. Although there results are not directly comparable to the ones presented in Table I due to the different training sets. It is, however, safe to state that the sequence-to-sequence method achieves similar accuracies as the state-of-the-art, if not higher.

However, the difference between the GE and OHE embeddings are not statistically significant. This can be explained by the simple letter-to-sound rules in Romanian which can be easily learned by this complex structure alone. It might be the case that for languages where the grapheme-to-phoneme conversion poses more complex problems, this difference could increase. The EL has slightly lower accuracy values, despite using twice the number of training epochs. This result can be explained by the fact that the EL might need more training data for the additional weights of the network. It is also interesting to note the fact that the syllabification and the lexical stress do not add that much value to the output accuracy. And also that the syllabification alone can reduce it. Again, it might be valuable to apply the same strategy to a more complex G2P language, as the Romanian results might already be plateaued.

As baseline, a simple decision tree classifier was also evaluated. The input to the decision tree is a window of graphemes centered around the predicted grapheme. Results for this algorithm are presented in Table VI for phone- and word-level accuracies, and with the addition of the linguistic information, i.e. syllabification and lexical stress. The feature encoding for the decision tree follows the steps described in [21]. Results show that even with this simple algorithm, the G2P task for Romanian can be solved with rather high accuracy.

TABLE V
ACCURACY RESULTS FOR **SEQ2SEQ** WITH **EMBEDDING LAYER**

	Accuracy [%]	
Input features	**Phone**	**Word**
Orthographic	98.75	95.40
Orthographic+Syllabification	98.42	95.26
Orthographic+Lexical stress	99.01	96.00
Orthographic+Syllabification+Lexical stress	99.15	96.25

TABLE VI
ACCURACY RESULTS OF THE **DECISION TREE CLASSIFIER** FOR A WINDOW LENGTH OF **5 CHARACTERS**

	Accuracy [%]	
Input features	**Phone**	**Word**
Orthographic	99.54	96.71
Orthographic+Syllabification	99.52	96.68
Orthographic+Lexical stress	99.56	96.95
Orthographic+Syllabification+Lexical stress	**99.60**	**97.17**

IV. CONCLUSIONS

This paper evaluated the use of sequence-to-sequence learning strategies for Romanian grapheme-to-phoneme conversion. The strategies involved the use of various input feature encodings, such as one-hot encoding, an additional embedding layer, or grapheme embeddings learned from an external text resource. The evaluation also included an analysis of combining the orthographic form of the words with their syllabification, lexical stress or both. The results of the systems were compared in terms of phone- and word-level accuracy scores. A decision tree algorithm trained to predict each phone individually from a character window sequence was included as baseline. The best results were obtained with the grapheme embeddings and all additional linguistic information, and achieve a phone-level accuracy of 99.62% and word-level accuracy of 97.90%. However, the difference between all the setups is not significant, and means that the sequence-to-sequence strategy is sufficient for the G2P task in Romanian.

One problem noticed in the sequence-to-sequence model prediction is that, because the decoder is conditioned only on the hidden state of the encoder, and that it only learns to predict the next phoneme in the output sequence, sometimes the order of the output phonemes is scrambled. One way to overcome this issue would be to condition the decoder on the current input grapheme as well, or to extend the available training data.

As future work, other network architectures could be considered. But it would also be interesting to test the simultaneous prediction of the phonetic transcription, syllabification and lexical stress assignment.

ACKNOWLEDGMENT

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] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards End-to-End Speech Synthesis," in *Proc. Interspeech*, 2017.

[2] W. Ping, K. Peng, A. Gibiansky, S. Ö. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep Voice 3: 2000-Speaker Neural Text-to-Speech," *CoRR*, vol. abs/1710.07654, 2017.

[3] D. Trandabat, E. Irimia, V. Barbu-Mititelu, D. Cristea, and D. Tufis, "The Romanian Language in the Digital Era," *Springer, Metanet White Paper Series*, 2012.

[4] V. B. Mititelu, E. Irimia, and D. Tufis, "CoRoLa:The Reference Corpus of Contemporary Romanian Language," in *LREC*, 2014, pp. 1235–1239.

[5] D. Tufiș, V. B. Mititelu, E. Irimia, Ș. D. Dumitrescu, and T. Boroș, "The IPR-cleared corpus of contemporary written and spoken Romanian language," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 2516–2521.

[6] The Romanian Explicative Dictionary (DEX) online. [Online]. Available: www.dexonline.ro

[7] A.-M. Barbu, "Romanian lexical data bases: Inflected and syllabic forms dictionaries." in *Proceedings of the International Conference on Language Resources and Evaluation*, 01 2008.

[8] R. Simionescu, "Graphical grammar studio as a constraint grammar solution for part of speech tagging," in *The Conference on Linguistic Resources and Instruments for Romanian Language Processing*, vol. 152, 2011.

[9] J. Domokos, O. Buza, and G. Toderean, "100k+ words, machine-readable, pronunciation dictionary for the romanian language," *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pp. 320–324, 2012.

[10] S.-A. Toma, A. Stan, M.-L. Pura, and T. Barsan, "MaRePhoR - An Open Access Machine-Readable Phonetic Dictionary for Romanian," in *Proceedings of the 9th Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, Bucharest, Romania, July, 6-9 2017.

[11] D. Burileanu, "Basic Research and Implementation Decisions for a Text-to-speech Synthesis System in Romanian," *International Journal of Speech Technology*, no. 5, pp. 211–225, 2002.

[12] D. Jitca, H. Teodorescu, V. Apopei, and F. Grigoras, "An ANN-based method to improve the phonetic transcription and prosody modules of a TTS system for the Romanian language," 01 2003.

[13] D. József, B. Ovidiu, and T. Gavril, "Automated grapheme-to-phoneme conversion system for Romanian," in *2011 6th Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, May 2011, pp. 1–6.

[14] T. Stefan-Adrian and M. Doru-Petru, "Rule-Based Automatic Phonetic Transcription for the Romanian Language," in *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, Nov 2009, pp. 682–686.

[15] M. A. Ordean, A. Saupe, M. Ordean, M. Duma, and G. C. Silaghi, "Enhanced Rule-Based Phonetic Transcription for the Romanian Language," in *2009 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, Sep. 2009, pp. 401–406.

[16] Ş. Toma, T. Birsan, F. Totir, and E. Oancea, "On letter to sound conversion for Romanian: A comparison of five algorithms," in *2013 7th Conference on Speech Technology and Human - Computer Dialogue (SpeD)*, Oct 2013, pp. 1–6.

[17] T. Boros, D. Stefanescu, and R. Ion, "Bermuda, a data-driven tool for phonetic transcription of words," in *Natural Language Processing for Improving Textual Accessibility (NLP4ITA) Workshop*, 2012.

[18] T. Boroș, "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*. Hissar, Bulgaria: INCOMA Ltd. Shoumen, BULGARIA, Sep. 2013, pp. 91–97.

[19] T. Boros, S. D. Dumitrescu, and S. Pipa, "Fast and accurate decision trees for natural language processing tasks," in *RANLP*, 2017.

[20] H. Cucu, A. Buzo, L. Besacier, and C. Burileanu, "SMT-based ASR domain adaptation methods for under-resourced languages: Application to Romanian," *Speech Communication*, vol. 56, pp. 195 – 212, 2014.

[21] A. Stan and M. Giurgiu, "A Comparison Between Traditional Machine Learning Approaches And Deep Neural Networks For Text Processing In Romanian," in *Proceedings of the 13th International Conference on Linguistic Resources and Tools for Processing Romanian Language (ConsILR)*, Jassy, Romania, November, 22-23 2018.

[22] C. Ungurean and D. Burileanu, "An advanced NLP framework for high-quality Text-to-Speech synthesis," in *2011 6th Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, May 2011, pp. 1–6.

[23] A. Stan, J. Yamagishi, S. King, and M. Aylett, "The Romanian speech synthesis (RSS) corpus: Building a high quality HMM-based speech synthesis system using a high sampling rate," *Speech Communication*, vol. 53, no. 3, pp. 442–450, 2011.

[24] T. Boros, S. D. Dumitrescu, and V. Pais, "Tools and resources for Romanian text-to-speech and speech-to-text applications," *CoRR*, vol. abs/1802.05583, 2018.

[25] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," *CoRR*, vol. abs/1506.00196, 2015.

[26] B. Peters, J. Dehdari, and J. van Genabith, "Massively multilingual neural grapheme-to-phoneme conversion," *CoRR*, vol. abs/1708.01464, 2017.

[27] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 4225–4229.

[28] B. Milde, C. Schmidt, and J. Kohler, "Multitask sequence-to-sequence models for grapheme-to-phoneme conversion," in *Proc. Interspeech 2017*, 2017, pp. 2536–2540.

[29] J. Ni, Y. Shiga, and H. Kawai, "Multilingual grapheme-to-phoneme conversion with global character vectors," in *Proc. Interspeech 2018*, 2018, pp. 2823–2827.

[30] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[31] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *CoRR*, vol. abs/1705.03122, 2017.

[32] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[34] C. Chelba, M. Norouzi, and S. Bengio, "N-gram language modeling using recurrent neural network estimation," Google, Tech. Rep., 2017.

[35] Z. Xie, "Neural text generation: A practical guide," *CoRR*, vol. abs/1711.09534, 2017.

[36] F. Deloche. Diagram for a one-unit Long Short-Term Memory (LSTM) - Wikimedia Commons. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/6/63/Long_Short-Term_Memory.svg

[37] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[38] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," 2016.

[39] O. Watts, "Unsupervised learning for text-to-speech synthesis," Ph.D. dissertation, University of Edinburgh, 2012.

[40] "Linguatools–Romanian Wikipedia database dump," https://linguatools.org/tools/corpora/wikipedia-monolingual-corpora/.

[41] L. van der Maaten and G. E. Hinton, "Visualizing High-Dimensional Data Using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[42] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Jul. 2011.