Turkish Journal of Electrical Engineering and Computer Sciences

Volume 30 | Number 6 Article 28

1-1-2022

Diacritics correction in Turkish with context-aware sequence to sequence modeling

ASİYE TUBA ÖZGE

ÖZGE BOZAL

UMUT ÖZGE

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 2433 – 2445
© TÜBİTAK
doi:10.55730/1300-0632.3948

Diacritics correction in Turkish with context-aware sequence to sequence modeling

Asiye Tuba KÖKSAL[1,2,*], **Özge BOZAL**[1], **Umut ÖZGE**[3]
[1]Huawei Turkey Research and Development Center, İstanbul, Turkey
[2]Department of Computer Engineering, Boğaziçi University, İstanbul, Turkey
[3]Informatics Institute, Middle East Technical University, Ankara, Turkey

Received: 15.11.2021 • **Accepted/Published Online:** 16.09.2022 • **Final Version:** 28.09.2022

Abstract: Digital texts in many languages have examples of missing or misused diacritics which makes it hard for natural language processing applications to disambiguate the meaning of words. Therefore, diacritics restoration is a crucial step in natural language processing applications for many languages. In this study we approach this problem as bidirectional transformation of diacritical letters and their ASCII counterparts, rather than unidirectional diacritic restoration. We propose a context-aware character-level sequence to sequence model for this transformation. The model is language independent in the sense that no language-specific feature extraction is necessary other than the utilization of word embeddings and is directly applicable to other languages. We trained the model for Turkish diacritics correction task and for the assessment we used Turkish tweets benchmark dataset. Our best setting for the proposed model improves the state-of-the-art results in terms of F1 score by 4.7% on ambiguous words and 1.24% over all cases.

Key words: Natural language processing, diacritics restoration, diacritics correction, sequence to sequence learning, LSTM

1. Introduction

Diacritics are orthographic manipulations indicating a change in the pronunciation of the letter they apply to. Most of the languages with Latin alphabet other than English (Croatian, Spanish, German, Romanian, Turkish, etc.) and many other languages with other alphabets (Arabic, Vietnamese, etc.) have diacritical letters. In languages like Turkish, where diacritics are used to distinguish between letters, a missing diacritic can either result in an out-of-vocabulary (OOV) word or cause an ambiguity problem, e.g., the diacritic-free surface form *aksam* is ambiguous between *aksam* ('component') and *akşam* ('evening').[1]

Ever growing use of mobile devices and social media has increased the spelling errors on the web especially those caused by ignoring diacritics while typing.[2] The transformation of diacritical letters to their ASCII counterparts is a trivial task, yet, the opposite transformation, namely diacritics restoration (DR), is an active subfield of text normalization, a vital prerequisite for various NLP pipelines.

Diacritics restoration, as the name implies, is usually seen as a task of restoring missing diacritics in words typed in ASCII. The present model aims to draw attention to the presence of examples in digital text which call

*Correspondence: tuba.koksal@boun.edu.tr

[1]Based on the counting of the 1 trillion tokens validated by TRmorph morphological analyzer [1] from the Oscar corpus [2], roughly 28% of Turkish words include diacritics, with 1.57 diacritics per word on average. Turkish diacritic letters and their ASCII counterparts are as follows: <ç,c>, <ş,s>, <ğ,g>, <ö,o>,<ü,u>, <ı,i>.

[2]Koksal et al. [3] report that with 44.94%, diacritic errors are the most common cause of misspellings in Turkish tweets.

for the opposite diacritic-to-ASCII conversion (see below for examples). Therefore, both transformations are required for proper text normalization. For this reason, we prefer to define the correction of erroneous use of diacritical letters as *diacritics correction (DC)* instead of diacritics restoration (DR). In languages like Turkish, DR is critical in resolving the ambiguity of the input text, as well. In this study, we aim not only to restore the missing diacritics but also to correct the misused ones in both ambiguous and unambiguous words.

Our contributions in this work can be summarized as follows:

- We formulate a context-aware character-level sequence to sequence model for the bidirectional transformation of the critical letters (from ASCII characters to diacritical counterparts and vice versa).
- The proposed model beats the state-of-the art model in terms of diacritics correction.
- Our model is better in inferring the correct forms of ambiguous words, so that it incites a change where and only where needed, without the aid of a language validator such as a morphological analyzer.

The paper is structured as follows: We give the related work in Section 2. We explain the methodology and the proposed model in Section 3 and the experimental setup in Section 4. We provide and discuss the results in Section 5 and finally conclude the paper in Section 6.

2. Related work

As mentioned above, since proper use of diacritics are important for NLP applications, there are many language specific [4–8] as well as language independent [9] models proposed in the literature. Initial studies apply statistical machine translation approaches for proper diacritics restoration.

Turkish specific studies which focus on converting ASCII characters to their diacritical forms, namely DR, date back to the early 2000s. The first published study addressing DR in Turkish [10] proposes a hidden Markov model which uses the state transition probabilities derived from a character-based language model. The Emacs Turkish mode program uses a decision list created on 1 million tokens collected from Turkish news by using a greedy prepend algorithm to correct the Turkish words written on English keyboard [11]. Okur et al. [12] compare different statistical methods such as N-grams and machine learning classifiers trained with word and sentence-level features concerning the ambiguity problem while restoring diacritics in Turkish. Yıldırım and Yıldız [13] generate all candidates including all variations of diacritics and their ASCII counterparts for the invalid Turkish words with the problematic letters. For this, the researchers initially check if there is a valid word in the candidate set and if there are multiple in-vocabulary (IV) candidates, then they select the most frequent one.

Turkish DR task has also attracted attention in information retrieval studies [14–16], since using only the ASCII versions of words may result in poor performance for languages with diacritical letters. The DR task is especially important in social media context where the users are more prone to error in the proper use of diacritics.

Adali and Eryiğit [17] adopt a two-level approach for Turkish diacritization task. The researchers firstly train a CRF model using character and token-level features to generate the most probable candidates and then use a morphological analyzer to validate them. Similarly, Ozer et al. [18] initially generate candidates for all the combinations of ASCII letters and their counterparts with diacritics. Word2Vec embeddings [19] are then trained with Turkish words to rank the candidates. If there are more than one valid word in the candidate set, the candidates are ranked according to the cosine similarities between the Word2Vec embeddings of the neighboring words and the candidate of interest.

More recent works, whether they propose language dependent or independent models, have been turning to deep learning approaches for diacritics restoration [20–22]. Primarily, they utilize character-level representations for correctly recovering the diacritics. For instance, Náplava et al. [23] propose one-for-all architecture for the diacritical languages, whereas Klyshinsky et al. [24] exhibit a detailed performance comparison of various neural network architectures over many diacritical languages. However, not all studies include contextual information into character representations. Ruseti et al. [25] incorporate fastText word embeddings together with the sentence embeddings encoded by a BiLSTM layer to capture semantic information of the context. Masala et al. [26] adopt similar approach by using BERT model's prediction on token embeddings of given sentence. Another language independent study [27] makes use of contextualized embeddings from pretrained BERT model by adding a feed-forward neural network with softmax output layer which gives a probability over the possible diacritics permutation given a diacritical word.

Some studies in Turkish diacritics restoration concentrate on character-level modeling, whereas others pay attention to the context information. Nonetheless, as far as we know, Turkish DR problem has not yet been approached by using character-level and context-related features together. On the other hand, although neural architectures with machine translation approaches such as sequence to sequence modeling have been proposed for DR problem in different languages [28–31], this approach has not been studied in Turkish so far.

We have been inspired by the architectures proposed and by [30] and [31]. Similarly, we formulate a character-based representation for the input and output sequences. Additionally, we aim to train a model which learns both character and context-level information for Turkish words. Therefore, distinctively from these studies, we formulate the input sequence as the character vectors of the source word concatenated with the word embeddings of its neighboring words. In this way, we make use of character and context-level information simultaneously.

To the best of our knowledge, this is the first work adopting a sequence to sequence modeling approach for diacritics correction in Turkish which utilizes character-level and contextual information at the same time to better handle ambiguous words.

3. Method

In this work, we propose a sequence to sequence model for DC problem in Turkish. Before providing the details of the model, we present some definitions that will be used throughout the paper.

We take a character to be critical in case there exist both diacritized and nondiacritized versions of it in the alphabet. This notion should be kept apart from being a diacritical character, which denotes a character that carries a diacritic manipulation. For instance, both *s* and *ş* are critical characters, but only the latter is a diacritical character. A word is a critical word, if it has at least one critical character.

We describe a word as incorrect if it needs to be altered to match the target, and as correct otherwise. A word is IV, only if it is a valid (i.e. grammatical) Turkish word. Throughout the study we use TRmorph[3], a finite-state rule-based morphological analyzer [1], to check for validity. Therefore, in our case, a word is IV as long as TRmorph generates a result for that word, and OOV otherwise. There are various reasons that makes a word OOV, but we are interested in only OOV words that are so due to a missing diacritic or a misused one, we simply ignore other types of misspellings.

It should be noted that being OOV implies incorrectness, but not the vice versa: There are incorrect IV words. One example in point would be as follows: Suppose a user intends to mean 'sour apple', which would

[3]https://github.com/coltekin/TRmorph [accessed 23 July 2021].

be rendered as *'ekşi elma'*, but types *'eksi elma'* instead, ignoring the diacritic on '*s*'. The word *eksi*, although incorrect in the sense that it differs from the target *ekşi*, is nevertheless a valid Turkish word, which means 'minus', hence IV. It, therefore, needs to be altered to *ekşi* resulting in *ekşi elma* ('sour apple').

3.1. Diacritization and dediacritization

One part of the task of diacritics correction is replacing misused ASCII characters with their correct diacritical counterparts, namely the task of diacritization. For instance, the first ASCII character *s* in the OOV word *semsiye* should be converted into the diacritical letter *ş* giving the IV word *şemsiye* ('umbrella'). However, we have observed that social media texts may contain examples which also require the opposite transformation —we name this dediacritization. As an example of dediacritization, the Turkish diacritical letter *ı* in the OOV word *kendım* should be restored to *i*, thereby transforming the given OOV word to the IV word *kendim* ('myself').[4] Additionally, in some situations diacritization and dediacritization may both be required. For instance, the OOV word *falçi* should be transformed to the IV word *falcı* ('fortune teller') where the letter *ç* is dediacritized to *c* and *i* is diacritized to *ı*.

We solidify the discussion in the above paragraph as follows: A word involves a diacritization error, in case at least one critical character in it must be *altered* to its diacritical form. Similarly, a word involves a dediacritization error, in case at least one critical character in it must be changed to its nondiacritic form. As the discussion above maintains, a word may involve both error types at the same time.

3.2. Ambiguous words

What we mean by ambiguity is explained in what follows. First, an auxiliary definition is that: Given a word w with at least one critical character, the diacritical family of w is the smallest set that includes every word w_j that can be formed only by replacing any number of the critical characters in w by their diacritized or nondiacritized counterparts. For instance the diacritical family of the word *eksi* is $\{eksi, ekşi, eksı, ekşı\}$, likewise the diacritical family of the word *saat* is $\{saat, şaat\}$. We take a word to be ambiguous, only if its diacritical family has more than one IV word, i.e.recognized by the morphological analyzer. In this respect, *eksi* is ambiguous, as there are two IV words, *eksi* and *ekşi*[5], in its family, while *saat* is not, as there is only one IV word in its family, namely itself.

It is worth at this point to emphasize a couple of points pertaining to major contributions of the present paper. When we focus on words that are incorrect due to diacritical reasons, it is a consequence of Turkish orthography that there exist words that are IV but incorrect —a class that overlaps in our case with what we take to be ambiguous. As these are IV, morphological analysis is helpless here and an effective utilization of contextual information is called for.

Another subtlety that arises due to ambiguous words is the following. It is quite likely to encounter critical words that are ambiguous and correct. A case in point is the phrase "*eksi* dokuz derece" ('*minus* nine degrees'), where *eksi* is such a word. It would have been a mistake, which we take care to avoid in the present work, to translate this word to its diacritized form *ekşi* ('sour'). As will be clarified in subsection 4.2, we put this point in view while assessing the performance of relevant models.

[4]This example gives the rationale for our definition of diacritization as 'orthographic manipulation' rather than 'adding extra marks': Here, the nondiacritic letter 'i' is being mapped to the diacritic version 'ı'.

[5]To be precise, as the concept of diacritical family defines an equivalence relation, all the members in a given family are ambiguous words, and therefore *ekşi* is also ambiguous in our terminology.

3.3. Sequence to sequence model

It is crucial to approach an NLP problem at the most appropriate level of representation (e.g., character, word, sentence, discourse etc.). Critical character errors occur at character-level. Moreover, word-level representations suffer from sparsity problems due to high number of OOV words which fall outside the limits of vocabulary size capped by limits on computational resources. Therefore, we approach the diacritics correction task as a character-level machine translation problem where we translate the wrongly used critical characters to their correct forms. For this translation problem we use a sequence to sequence model.

The objective of the model is two-fold: we aim firstly to correct misused critical characters, secondly to preserve the correct usage of them.

3.3.1. Input sequence

The input sequence is the character-level representation of words from the source sentences. Each character in an input sequence is represented by a d-dimensional one-hot-encoded vector, where d is the number of unique characters encountered in our training data. Then, each word is initially represented by a matrix of size $m \times d$ with padding, where each row stands for a character in the word and m is the maximum word length.

3.3.2. Context window

The correct forms of ambiguous words highly depend on the context as mentioned in subsection 3.2. Therefore, the character-level representation mentioned above is not sufficient on its own for the model to distinguish between the correct forms of those ambiguous words. For that matter we refer to their source sentences to extract the context information of input words. We utilize the context information by concatenating the word embeddings of the surrounding words in the context window to each one-hot-encoded character vector of the corresponding word, which means that for context window size n, each character in the input sequence is then represented with a $(d + n.e)$-dimensional vector, where e is the dimension of word embedding vectors.

The context words, however, may contain OOV words. For this reason word embeddings which are highly representative on OOV words are required.

3.3.3. Model

The model learns to map the character sequence of the source word w_i to that of the target word t_i conditioned on the character sequence of w_i itself and the context words $w_{i-n}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+n}$, where n is the context window.

While learning this mapping, first the encoder maps w_i in a source sentence encoded as a character sequence as described above to an internal representation, and the decoder takes as input this internal representation and the one-hot-encoded representation of the target t_i, this time without the context information concatenated. The decoder output is then carried to a softmax layer to generate a prediction.

(1) bu <u>aksam</u> gel

In the sample source sentence above (1), the input sequence *aksam* ('component') is ambiguous in the sense that it should be changed to *akşam* ('evening') or it should be left as it is, depending on the context. Assuming a context window size $n = 1$, the ambiguity can be resolved by taking into account the nearby words *bu* ('this') and *gel* ('come'). This way, the target word is changed to *akşam*. As depicted in Figure (1), the encoder is fed with one-hot encoded character vectors of *aksam* each concatenated with the word embedding vectors of its surrounding words *bu* and *gel*. The decoder, on the other hand, accepts only the character vectors of *akşam* as input.

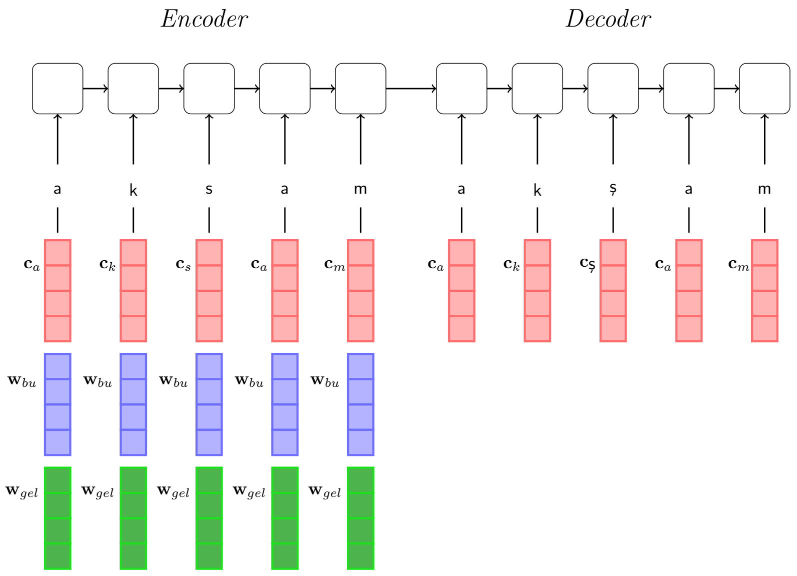

Figure. Input sequences of encoder and decoder blocks. The example input sequence corresponding the input word *aksam* when the context window size $n = 1$. The one-hot encoded character vectors, word embedding vectors for the context word *bu* and *gel* are denoted as $\{c_a, c_k, c_s, c_a, c_m\}$, w_{bu} and w_{gel}, respectively.

4. Experimental setup

In this section, we provide a description of our experimental setup based on the proposed method as defined in subsection 3.3. We discuss in order, the details of the training and evaluation phases, the experimental settings, and the baseline models used for comparison.

4.1. Training

We prepared our training and validation datasets using the Oscar corpus [2]. First, we discarded the sentences with at least one word not recognized by the morphological analyzer. All characters were transformed to lowercase to reduce the number of unique input characters, thereby reducing the complexity of the network. Finally, punctuation characters except for the word-internal ones were removed. Following this filtering and preprocessing, we randomly collected sentences from the corpus for training and validation sets. The statistics of the datasets are given in Table 1.

In real user text, sentences usually contain more than one word with critical character errors. For this reason, having an incorrect word with correct context words would not be a realistic assumption for training. Therefore, to mimic real user scenarios, we manipulated source sentences such that each has both proper and improper uses of critical characters as in (2), the correct form of which is (3).

(2) falçinin tahmınlerı dogru cıkarsa eksi elma yiyecegım

(3) falcının tahminleri doğru çıkarsa ekşi elma yiyeceğim
('If the fortune teller is right, I will eat sour apples.')

Table 1. Train and validation dataset statistics.

	Train	**Validation**
Unique characters	52	52
Critical characters	84.4 M	16.9 M
Wrong critical characters	42.2 M	8.4 M
Total words	42.6 M	8.5 M
Unique words	2.2 M	0.9 M
Sentences	3.66 M	0.7 M

We manually introduced these errors into the training input for the encoder and we used the error-free versions in the decoder. The validation input was also manipulated by the same method. There are 84.4 M critical characters in the training set and half of them, which were randomly selected, were replaced with their erroneous counterparts.

In this setting the encoder and decoder input differ only with respect to their critical characters. As described in subsection 3.1, we aim to make a 2-way correction, therefore we added both diacritization and dedeacritization errors.

4.2. Evaluation

For the assessment of the proposed model, we used the annotated benchmark Turkish tweets dataset [3]. The error category annotated as *OOV-ill_formed-deascii* in this dataset consists of diacritization and dediacritization cases, which serves as an appropriate test dataset for the proposed model.

We selected 1165 preprocessed sentences from the Turkish tweets dataset that contain at least one incorrect word due to critical character errors. We then replaced all other misspelled words —those that are OOV due to other reasons— with their correct forms.[6] This way, we modified Turkish tweets dataset such that it consists of incorrect words with critical character errors and IV words. The statistics of the modified test dataset can be seen in Table 2.

Table 2. Word counts in test dataset.

	Word count
Incorrect (all)	2522
No diacritic	2221
Mixed	301
Incorrect (all)	2522
OOV	1964
IV	558
Ambiguous	1522
Incorrect	558
Correct	964

It is crucial to observe that the present scheme affords two different ways of breaking down incorrect

[6]Note that we crucially diverge here in terminology from Koksal et al. [3], they would take the word *eksi* in our example as OOV due to diacritical reasons.

words into subcategories. On one hand we distinguish incorrect words that consist of only ASCII characters (Incorrect - no diacritic) from those that contain at least one diacritical character (Incorrect mixed); on the other hand we distinguish words that are incorrect and OOV (Incorrect OOV) from those that are incorrect, but nevertheless IV (Ambiguous incorrect).

Please note that for all the evaluations, we provided full sentences to each model to retrieve context information. Nonetheless, we evaluated the model performances on word-level in terms of the evaluation metrics discussed shortly below.

Our comparative evaluation of the relevant models involves the following metrics: *precision*, *recall*, *F1* score and *noncorruption* rate as formulated in Equations (1)–(4), respectively.

$$Precision = \frac{\text{\# of corrected words}}{\text{\# of changed words}} \tag{1}$$

$$Recall = \frac{\text{\# of corrected words}}{\text{\# of words that need correction}} \tag{2}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

$$Noncorruption = \frac{\text{\# of unchanged correct words}}{\text{\# of correct words}} \tag{4}$$

Recall that the paper aims to draw attention to two aspects of the task of diacritics correction: one concerning the bidirectional nature of diacritics correction (subsection 3.1), and the other concerning the ambiguity problem (subsection 3.2). In order to better capture each model's performance regarding these aspects, we designed a two-stage evaluation:

First, we compared the models along their precision and recall performance on all incorrect words in the test dataset. We further split the incorrect words into two mutually exclusive and exhaustive subsets: (i) words with only diacritization errors, (ii) words with only dediacritization errors and words with both types of errors. Note that this categorization is necessary to segregate the error types described in subsection 3.1 and does not make a distinction between the incorrect ambiguous and incorrect OOV words.

Second, we formulated a separate evaluation to assess each model's performance on ambiguous words described in subsection 3.2: We selected the correct and incorrect ambiguous words (Ambiguous) and evaluated two aspects of these ambiguous words: (i) correction of incorrect ambiguous words, and (ii) noncorruption of correct ambiguous words.

We provide precision, recall, F1 score, and noncorruption results computed separately on the above datatests in Section 5.

4.3. Baseline models

As a baseline for performance comparisons we take the setup where word embeddings are used to filter candidates generated in a rule-based fashion. In Turkish DR literature we only encounter the study of Ozer et al. [18] which incorporates word embeddings. The researchers first generate the diacritic family of a critical word. Then they select the valid Turkish words among them by a morphological analyzer-based validator. For the cases where more than one candidate are valid, the model should select semantically the most appropriate one. They train general purpose Turkish Word2Vec embeddings for this purpose and use them to distinguish between correct forms of the ambiguous words. They compare the cosine similarities of surrounding words' Word2Vec embeddings with that of each candidate and select the candidate with the highest cosine similarity score as the correct form of the incorrect word.

Similarly, we design such a correction mechanism with fastText[7] embeddings proposed by Bojanowski et al. [32] and BERT[8] pretrained embeddings from the community-driven project BERTurk[9] to provide a baseline for the proposed model. We specifically prefer fastText and BERT embeddings since they are also representative on OOV words presumably thanks to being trained on subwords as well as words themselves. We validate the candidates with the morphological analyzer. We then compute the similarity score for each candidate as the average of cosine similarities of corresponding word embedding vectors of each context word and the candidate word, and select the candidate with the highest similarity score.

We compare the results of these baseline models in Tables 3 and 4 on the final dataset described in subsection 4.2. We then utilize the embeddings with the highest score in the proposed model for training purposes.

Table 3. Precision, recall and F1 score (in %) of the models on (1) all incorrect words (2) incorrect words with only ASCII characters that need diacritization, (3) *miscellaneous* category defines the union of incorrect words containing only dediacritization errors and those having combination of both diacritization and dediacritization errors.

	All incorrect			**Diacritization**			**Miscellaneous**		
Model	**Pr**	**Rc**	**F1**	**Pr**	**Rc**	**F1**	**Pr**	**Rc**	**F1**
ITU	98.66	96.63	97.63	98.71	96.49	97.59	98.32	97.67	97.99
Emacs TM	98.71	97.02	97.86	98.81	96.89	97.84	98.0	98.0	98.0
Zemberek	98.63	96.83	97.72	98.76	96.76	97.75	97.66	97.33	97.49
fastText	97.33	95.52	96.42	97.39	95.72	96.55	96.91	94.0	95.43
BERT	96.30	90.76	93.44	96.50	90.59	93.45	94.85	92.0	93.40
T1 + Char	99.08	97.86	98.47	99.09	97.84	98.46	98.99	98.0	98.49
T2 + Char	99.0	97.98	98.49	98.95	97.97	98.46	99.32	98.0	98.65
T1 + Char + 1n	99.48	98.73	**99.10**	99.50	98.69	**99.09**	99.33	99.0	**99.16**
T1 + Char + 2n	99.28	98.37	98.82	99.27	98.33	98.80	99.33	98.67	99.0
T2 + Char + 1n	99.20	98.49	98.84	99.23	98.42	98.82	99.0	99.0	99.0
T2 + Char + 2n	99.36	98.45	98.90	99.41	98.38	98.89	99.0	99.0	99.0

For a much better evaluation of the effect of utilizing contextual information in character representation, we trained another baseline model without using the contextual information in character representation, i.e.we trained the proposed model with the context window size $n = 0$. We denote this model as *Char* in the tables provided in Section 5.

4.4. Experimental settings

We use unidirectional 2-layer LSTM for the encoder and a single layer unidirectional LSTM for decoder blocks. For training the models we used batch size = 128, hidden layer size = 512, Adam optimizer with the learning

[7]https://fasttext.cc/docs/en/crawl-vectors.html [accessed 15 May 2021].
[8]https://huggingface.co/loodos/bert-base-turkish-uncased/tree/main [accessed 29 December 2021].
[9]https://zenodo.org/record/3770924#.YfQLehdRVhE [accessed 29 December 2021].

Table 4. Precision, recall, F1, and noncorruption rate results (in %) of the models on all incorrect and correct ambiguous words.

Ambiguous words (correct and incorrect)				
Model	**Pr**	**Rc**	**F1**	**Nc**
ITU	84.35	88.89	86.56	92.74
Emacs TM	89.16	92.83	90.96	94.81
Zemberek	86.59	91.40	88.93	93.36
fastText	75.74	86.74	80.87	89.11
BERT	49.08	71.68	58.27	64.21
T1 + Char	84.78	92.83	88.62	91.91
T2 + Char	84.85	93.37	88.91	92.01
T1 + Char + 1n	94.41	96.95	**95.66**	**97.20**
T1 + Char + 2n	93.01	95.34	94.16	97.0
T2 + Char + 1n	93.23	96.24	94.71	96.99
T2 + Char + 2n	84.85	93.37	88.90	92.01

rate of 10^{-4} and dropout of 0.2 in all of the LSTM layers. For the context-aware character representations of input sequences we used the fastText word embeddings as they give better results in diacritics correction in all the test cases. We trained the models with 2 different input sequence formulations: with context window size $n = 1$ and 2, denoted as *Char + 1n* and *Char + 2n*, respectively, in the tables provided in Section 5.

The source sentences in the training dataset contain words that are not in the scope of DC such as the word *araba* ('car'), which contains no critical characters. Nevertheless, we used such words in the training, since we conjectured that they would be helpful in learning character sequences. To see the effect of such words in the training, we performed ablation experiments with various context window sizes, where we excluded such words from the training data. Please note that we differentiate the models in Section 5 according to their training dataset by denoting the models trained on all words with *T1* and on the second dataset with *T2*.

5. Results and discussion

In this section we provide the results of the proposed model and baseline models described in subsection 4.3. We also compare our results with the following publicly available DR tools: ITU Turkish Natural Language Processing Pipeline's deasciifier (ITU)[10] developed by Adali and Eryiğit [17], Emacs Turkish mode program (Emacs TM)[11], and the deasciifier in Zemberek-NLP tool for Turkish[12], which is inspired by Emacs Turkish mode program.

The present model aims to solve the diacritization and dediacritization errors explained in subsection 3.1 in all incorrect words. We provide the precision, recall and F1 scores of the models over all incorrect words under the heading 'All incorrect' in Table 3. Recall that one of the main arguments of this paper is that the diacritics restoration task should better be thought of as diacritics correction, which implies that we need to take care of dediacritization as well as diacritization. To better differentiate the models' performance regarding this distinction, we computed the scores by using different word categories as explained in subsection 4.2: (i)

[10] http://tools.nlp.itu.edu.tr/Deasciifier [accessed 05 June 2021]
[11] http://www.denizyuret.com/2006/11/emacs-turkish-mode.html [accessed 05 June 2021]
[12] https://github.com/ahmetaa/zemberek-nlp [accessed 08 May 2021]

diacritization errors on one hand, and (ii) dediacritization and mixed type of errors on the other. The results are given under the 'diacritization' and 'miscellaneous' headings in Table 3, respectively.

We describe the proposed model as context-aware such that it can distinguish between the correct forms of the ambiguous cases described in subsection 3.2. Namely, the model corrects the incorrect ambiguous words while leaving intact the correct ones. We give the precision, recall, F1 and noncorruption rates of the models only on ambiguous words in Table 4.

- In all evaluation scenarios the best performing model in F1 and noncorruption is the one with context window size $n = 1$ trained on *T1* dataset.
- The context-unaware baseline models *T1 + Char* and *T2 + Char* are also better than the state-of-the-art in terms of diacritization and dediacritization of incorrect words overall. However, they fall behind the context-aware models (both T1 and T2) in ambiguous words as expected. Character-level representation without contextual information alone can beat the state-of-the-art over unambiguous cases; however, the context information is indispensable for ambiguous cases.
- *Char+2n* models fall behind *Char+1n* models. One reason could be that a larger context window size, by increasing the input dimension, might have suppressed the character-level information within the input.
- Lastly, our hypothesis for the ablation experiments was that noncritical words included in the training process increase the capacity of the model in representing the character-level information, therefore boost the performance in diacritics correction. The proposed model trained on *T1* performs better than that the one trained on *T2*. For that matter our hypothesis is proven to be effective.

6. Conclusion

The paper motivated and empirically supported a sequence to sequence model architecture for diacritics correction task in Turkish. Our model brought a two-way perspective on diacritics correction, in modeling both nondiacritic to diacritic and the opposite transformations. We showed that the proposed model enjoys above state-of-the-art performance in domains with various information content regarding Turkish diacritics correction task.

Although our assessments were based on Turkish data, the proposed model is language independent, since it does not make use of any language dependent features, and therefore would be applicable to languages where diacritics serve the same function they do in Turkish.

We plan the following extensions and modifications of the present system as a future work: (i) having a trainable dense embedding layer for character-level input; (ii) fine tuning the context word embeddings and/or learning these embeddings from scratch jointly with the task; (iii) extending the DC system to full-fledged text normalization.

References

[1] Çöltekin Ç. A freely available morphological analyzer for Turkish. In: The Seventh International Conference on Language Resources and Evaluation (LREC'10); Valletta, Malta; 2010. pp. 820–827.

[2] Suárez PJO, Romary L, Sagot B. A monolingual approach to contextualized word embeddings for mid-resource languages. In: The 58th Annual Meeting of the Association for Computational Linguistics; Jeju Island, Korea; 2020: 1703–1714

[3] Koksal AT, Bozal O, Yürekli E, Gezici G. #Turki$hTweets: A benchmark dataset for Turkish text correction. In: Findings of the Association for Computational Linguistics: EMNLP 2020; Online; 2020. pp. 4190–4198.

[4] Masmoudi A, Mdhaffar S, Sellami R, Belguith LH. Automatic diacritics restoration for Tunisian dialect. ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP) 2019; 18 (3): 1–18.

[5] Novák A, Siklósi B. Automatic diacritics restoration for Hungarian. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing; Lisbon, Portugal; 2015. pp. 2286–2291.

[6] Azmi AM, Almajed RS. A survey of automatic Arabic diacritization techniques. Natural Language Engineering, 2015; 21 (3): 477–495.

[7] Nguyen KH, Ock CY. Diacritics restoration in Vietnamese: letter based vs. syllable based model. In Pacific Rim International Conference on Artificial Intelligence; Daegu, Korea; 2010. pp. 631–636.

[8] Yarowsky D. A comparison of corpus-based techniques for restoring accents in Spanish and French text. In: Armstrong S, Church K, Isabelle P, Manzi S, Tzoukermann E et al. (editors). Natural Language Processing Using Very Large Corpora. Dordrecht: Springer, 1999, pp. 99–120.

[9] Mihalcea R, Nastase V. Letter level learning for language independent diacritics restoration. In: COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002); Philadelphia PA; 2002.

[10] Tür G. A statistical information extraction system for Turkish. PhD, Bilkent University, Ankara, Turkey, 2000.

[11] Yuret D, De La Maza M. The greedy prepend algorithm for decision list induction. In: International Symposium on Computer and Information Sciences; Istanbul, Turkey; 2006. 37–46.

[12] Okur BÇ, Takçi H, Akgül YS. Rewriting Turkish texts written in English alphabet using Turkish alphabet. In: 21st Signal Processing and Communications Applications Conference (SIU); Haspolat, Turkey; 2013. pp. 1–4.

[13] Yıldırım S, Yıldız T. An unsupervised text normalization architecture for Turkish language. Research in Computing Science 2015; 90: 183–194.

[14] Alpkocak A, Ceylan M. Effects of diacritics on Turkish information retrieval. Turkish Journal of Electrical Engineering & Computer Sciences 2012; 20 (5): 787–804.

[15] Çakmak F, Diri B. Correction of Turkish characters with a web-based semantic method. In: 23nd Signal Processing and Communications Applications Conference (SIU); Malatya, Turkey; 2015. pp. 891–894.

[16] Arslan A. Deasciification approach to handle diacritics in Turkish information retrieval. Information Processing & Management 2016; 52 (2): 326–339.

[17] Adali K, Eryiğit G. Vowel and diacritic restoration for social media texts. In: The 5th Workshop on Language Analysis for Social Media (LASM); Gothenburg, Sweden; 2014. pp. 53–61.

[18] Ozer Z, Ozer I, Findik O. Diacritic restoration of Turkish tweets with word2vec. Engineering Science and Technology, an International Journal 2018; 21 (6): 1120–1127.

[19] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: The 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13; Red Hook, NY, USA; 2013. pp. 3111–3119.

[20] Nga CH, Thinh NK, Chang PC, Wang JC. Deep learning based Vietnamese diacritics restoration. In: IEEE International Symposium on Multimedia (ISM); San Diego, CA, USA; 2019. pp. 331–3313

[21] Hucko A, Lacko P. Diacritics restoration using deep neural networks. In: IEEE World Symposium on Digital Intelligence for Systems and Machines (DISA); Kosice, Slovakia; 2018. pp. 195–200.

[22] Alkhatlan A, Kateb F, Kalita J. Attention-based sequence learning model for Arabic diacritic restoration. In: IEEE 6th Conference on Data Science and Machine Learning Applications (CDMA); Online; 2020. pp. 7–12.

[23] Náplava J, Straka M, Straňák P, Hajic J. Diacritics restoration using neural networks. In: The Eleventh International Conference on Language Resources and Evaluation (LREC 2018); Miyazaki, Japan; 2018. pp. 1566–1573.

[24] Klyshinsky E, Karpik O, Bondarenko A. A comparison of neural networks architectures for diacritics restoration. In: IEEE International Conference on Analysis of Images, Social Networks and Texts; Moscow, Russia; 2020. pp. 242–253.

[25] Ruseti S, Cotet TM, Dascalu M. Romanian diacritics restoration using recurrent neural networks. arXiv preprint arXiv:2009.02743; 2020.

[26] Masala M, Ruseti S, Dascalu M. Robert–a Romanian BERT model. In: The 28th International Conference on Computational Linguistics; Barcelona, Spain; 2020. pp. 6626–6637.

[27] Náplava J, Straka M, Straková J. Diacritics restoration using BERT with analysis on Czech language. arXiv preprint arXiv:2105.11408; 2021.

[28] Orife I. Attentive sequence-to-sequence learning for diacritic restoration of Yoruba language text. arXiv preprint arXiv:1804.00832; 2018.

[29] Hung BT. Vietnamese diacritics restoration using deep learning approach. In: IEEE 10th International Conference on Knowledge and Systems Engineering (KSE); Ho Chi Minh City, Vietnam; 2018. pp. 347–351.

[30] Nuţu M, LℍOrincz B, Stan A. Deep learning for automatic diacritics restoration in Romanian. In: 15th International Conference on Intelligent Computer Communication and Processing (ICCP); Cluj-Napoca, Romania; 2019. pp. 235–240.

[31] Mubarak H, Abdelali A, Sajjad H, Samih Y, Darwish K. Highly effective Arabic diacritization using sequence to sequence modeling. In: 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1; Minneapolis, Minnesota, USA; 2019. pp. 2390–2395.

[32] Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606; 2016.