Turkish Diacritization with Deep Learning

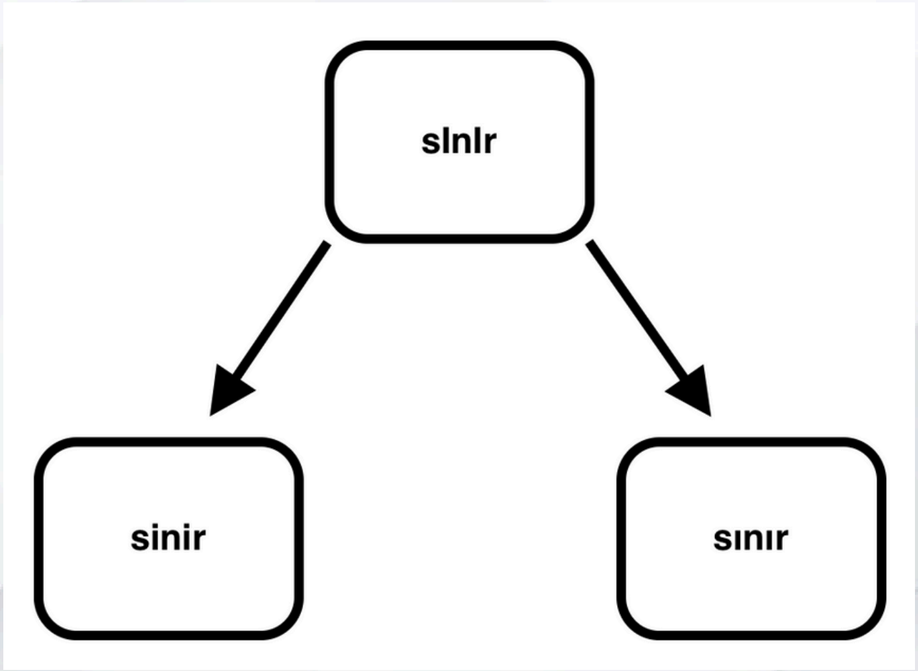
Emirberk Almacı
Furkan Öztürk

22.05.2024

Diacritization

Diacritics are essential marks added to letters, altering their pronunciation. They are utilized in various languages like Turkish, Arabic, Greek and Hungarian. However, keyboard constraints and individual preferences led to the widespread adoption ASCII equivalents, relegating accented letters to secondary characters. For example, Turkish consists of seven diacritic characters (ç, ı, l, ğ, ö, ş, ü) and their ASCII equivalents are (c, i, l, g, o, s, u), respectively.

The correct diacritization of Turkish text is crucial for various natural language processing tasks, including machine translation, automatic post-editing, language modeling, and speech recognition.



Existing Research

Maximum Entropy

Estimating the probability distribution over the possible diacritics for each word in the input text (Zitouni and Sarikaya, 2009).

CRF

Best sequences are produced at the character level by using CRF.
Afterwards, the one with the highest probability is selected with Language Validator and produced as output (Adali and Eryiğit, 2014).

RNN

It is seen that deep learning models, especially LSTM models, are widely used in RNN structures (Madhfar and Qamar, 2021). Additionally, in this study, studies conducted with encoder - decoder and decoder-only models reached a higher success rate.

BERT

The study demonstrates the effectiveness of leveraging BERT's contextual understanding to accurately predict missing diacritics, outperforming traditional rule-based and statistical methods. In addition to the Czech language, the model was also applied to Turkish and good results were obtained (Náplava et al., 2021).

Dataset

Dataset

Train.csv (Provided):

- krep hamuru bitene kadar işleme devam edilir
- büyük olasılıkla zararsız ama tartışmalı

Turkish Data Depository(data.tdd.ai):

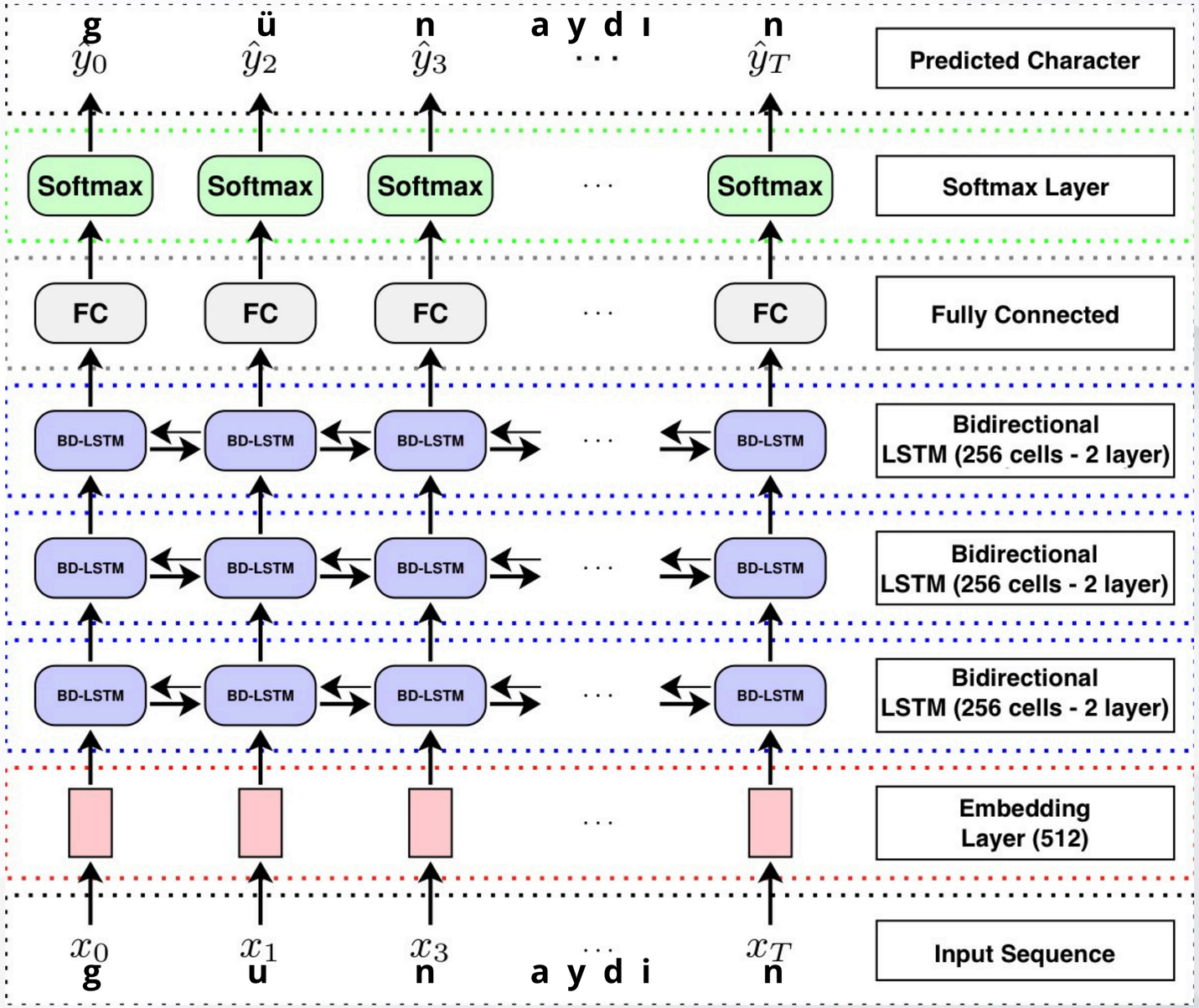
- direnimlere
- neticelerinin

Tools

- Numpy
- Pandas
- PyTorch

Challenges

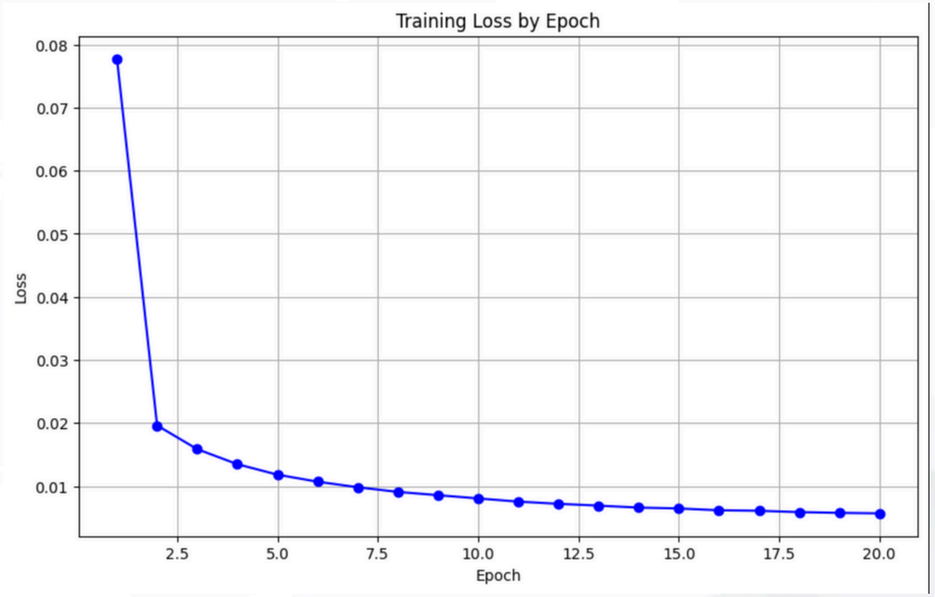
Model Structure



(It is assumed that no padding is applied.)

Train Process

Cross – Entropy Loss



$$L = -\sum y_0 log(predict)$$

Adam Optimizer

- Learning rate = 0.001
- Weight decay = 0.01

Hyperparameters

- Batch size = 64
- Embedding dim = 512
- Hidden size = 256
- Vocab size = index_count + 1
- N classes = index_count + 1
  (+ 1 for not valid char)

Results

92%
Accuracy on test set

Pros

- The result obtained for the RNN structure is good, although improvements can be made.
- The dataset was trained with many words. Therefore, it is robust against different word types.

Cons

- The error rate is higher for words whose context or meaning changes significantly depending on other words in the sentence.
- Model can be handle complex patterns and long-range dependencies more effectively.

Summary and Future Research

Although the model performs well, it can achieve better results with improvements and support from other models. Success rates of around 98% in studies support this situation.

For Future

- **Attention**

It can also be used to achieve high success in words whose context or meaning changes significantly depending on their use with other words in the sentence.

- **Different Model (like Encoder-Decoder)**

Newer structures can be used to help the model handle long-term dependencies more effectively.

Thanks for Listening