YZV 405E NLP Term Project Proposal

Bora Boyacıoğlu
Istanbul Technical University
Artificial Intelligence and Data Engineering
150200310
boyacioglu20@itu.edu.tr

Abstract

This document is for an NLP project, aiming to fix character dialetics happening in a variety of similar languages. The process, called diacritization, will be done using an NLP model. I am, the author, Bora Boyacıoğlu, a student in Istanbul Technincal University. This is my project proposal for the specified task.

1 Introduction

With the effects of globalisation, languages began to surrender their unique features, especially their alternative letters, to English. English is considered to not have accented characters, as it is used as a base language. Also considering computer era raised in English-speaking world (USA, UK, Canada), the main character set used in the informatics (ASCII, American Standard Code for Information Interchance) is English based. Therefore Turkish and many other European languages being Latin Alphabet based, there are still very common characters which not been represented in the ASCII character set.

ASCII is used widely. From the internet addresses to SMS's (Short Message Service), a lot of systems used by billions of people used daily are only compatible with ASCII charset, with some like SMS only supporting a limited extra set in its essence (ISO 8859-1). This means people actually use non-decorative characters to write their more complex languages. This creates an issue. While a lot of words being understood easily even when using similar but different letters, some get confused easily. To give an example, Turkish has "oldum" (I have been) and "öldüm" (I was died), which are both represented as "oldum" in the ASCII set.

With the new context-based understanding applications, this is tried to be overcome using NLP techniques. In this project, my goal is too, do a similar task. I will explain the problem in detail. Though before, it is necessary to list a few example works have been done in other projects.

2 Dataset

In this task, I am using the project dataset, which is available on this Kaggle Competition. There are two data available initially: a train and a test data in CSV (Comma-Separated Values) format. These are essentially two column data where the first column reserved for the entry ID. The second, and the main column, contains Turkish sentences written in non-Turkish specific ASCII character set, in the test data. To train the model, training data provides me with the real full-charset Turkish sentences. There are 1145 sentences available in the test data. To give an example of the non-Turkish charset, in Table 1 a couple of sentences from the test data are shown.

18	diyalog icin de kapi acildi aclik grevle...
24	Israil cocuklara acimiyor Erdogan Kah...
34	kadini cahil tutmak suretiyle somurme...
55	isim secimimin otobiyografik nedenle...

Table 1: Example Sentences from Test Data

On the other side, the train data is completely normal structured, with proper letters and fully meaningful words. This will be a necessity as my model will be trained over this data. There are 40986 sentences to train the model in the train data, which is hopefully more than enough. To give an example from the training data, here are a few sentences in Table 2.

25	isterseniz hızlı arama veya detaylı ara...
46	bu iki yemek tarifimiz girit yemekleri...
66	krep hamuru bitene kadar işleme deva...
99	ilk önce hayatsal fonksiyonları neden...

Table 2: Example Sentences from Train Data

Ther are a few characters different betweeen the plain-English ASCII and ISO 8859-9 Turkish sets: native Turkish doesn't have *q/Q, w/W, x/X* char-

acters while ASCII not having *ç/Ç, ğ/Ğ, ı, İ, ö/Ö, ş/Ş, ü/Ü*. Of course, absence of some characters in the native Turkish isn't much of a problem since this data is already Turkish and all the charsets supporting English characters. However, there are actually 12 letters including the capitals which not have been represented in the given test dataset.

My main goal is trying to overcome this issue by utilizing an NLP model, which learns from the train set and will be hopefully able to work on the test set. This is, *diacritize* the words back to their original form with their original letters.

3 Related Work

There are a number of studies conducted for this task. Interestingly, most of them are specifically done for Arabic. Though, the fact that their alphabet is different than both Turkish and English, I would not prefer to base my project on an Arabic study. Instead, there are also less popular but important studies done in European languages.

One of the studies I found really interesting is "Diacritics Restoration using BERT with Analysis on Czech language" Náplava et al. (2021). Although it is based on Czech and conducted in Czech Republic, it actually includes many European languages, some more extensively than Czech. And the fact that their instruction set is the same in both Turkish and Czech data highlights its importance for this project. With 1005 as their Turkish set size and 10884 in total, they get around 98.98% accuracy for Turkish text. This is really useful for me. And even though they are utilizing a pre-trained BERT model, there are still lots of important points I will be inspiring from.

There is an other article I want to point out. It is "Diacritics correction in turkish with context-aware sequence to sequence modeling" Köksal et al. (2022). The beauty is, it is done by Turkish people and focused on the problem Turkish specific case. Also, their data includes more than 50 million words in total. This shows that the results will be more satisfying. They rely on a couple of models, one of them being ITU. It is a **deasciifier** model develop by Eryiğit (2014). Their different models combined, have more than 90% F1 scores.

Lastly, because its variety, it is crucial to mention one of the Arabic diacritization studies. I want to focus on the study "Arabic Diacritization with Recurrent Neural Networks" (Belinkov and Glass, 2015). Their approach invlolves an RNN to solve more complex problem, as they are focusing on non-Latin languages like Arabic and Hebrew. Their involvement of more than 600k words in total, and utilization of a strong model, will be very beneficial for me.

4 Model Selection

There are a variety of models may be used in a diacritization NLP task. Some will be very helpful for me, while some will just guide me on what can be done. First, I would like to list a couple of pre-trained models.

4.1 Auto-Trained Models

At the top of the list, there is BERT. It is powerful as it understands the meaning behind the words, and this really creates a difference compared to traditional rule based systems. Also, its bidirectionality allows extracting the meaning from the sentence.

There are also variaties of BERT, like RoBERTa, Mulilingual Bert and TurkuNLP. These will improve performance, especially the ones supporting Turkish language. However, the goal here is to train a model from scratch. So, let me focus on other models.

4.2 Seq2Seq

Sequence-to-Sequence (Seq2Seq) models are a framework consisting of an encoder and a decoder parts. The encoder processes the sequence into a vector, and the decoder uses this vector to generate an output sequence.

They may actually be useful for a task like diacritization as with their availability to work with variety-length of inputs. Here, the input will be the ASCII text and the output will be the final, true form.

4.3 RNN

Recurrent Neural Networks (RNNs) are designed for sequential data processing, as well. They keep the previous inputs their "memory", which allows them to carry the information onto the next tasks.

However, due to the issues like vanishing and exploiding gradients, they may struggle with long-range dependancies. This makes them less efficient for longer sentences.

4.4 LSTM

Long Short-Term Memory (LSTM) Units are an advanced type of RNNs, solving the previously

mentioned vanishing and gradient exploiding issues. They do this by a complex structure of gates that regulate the flow of information, deciding what to keep or discard from the memory.

With the problems of RNNs solved, LTSMs can more effectively work on diacritization tasks. After training on an extensive dataset, they recongize the patterns and decide on what to keep on the memory.

References

Yonatan Belinkov and James Glass. 2015. Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2281–2285.

Gülşen Eryiğit. 2014. ITU Turkish NLP web service. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, Sweden. Association for Computational Linguistics.

Asiye Tuba Köksal, Özge Bozal, and Umut Özge. 2022. Diacritics correction in turkish with context-aware sequence to sequence modeling. *Turkish Journal of Electrical Engineering and Computer Sciences*, 30(6):Article 28.

Jakub Náplava, Milan Straka, and Jana Straková. 2021. Diacritics restoration using BERT with analysis on czech language. *CoRR*, abs/2105.11408.