DIACRITIC RESTORATION OF TURKISH SENTENCES

(TÜRKÇE CÜMLELERDE FONETİK İŞARETLERİN DÜZELTİLMESİ)

by

Hüseyin Ekici, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

Oct 2021

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my esteemed supervisor, Assoc. Prof. İ. Burak Parlak, for his unwavering support and belief in me.

I am deeply grateful to my family and friends for their encouragement and support all through my studies.

June 2021
Hüseyin EKİCİ

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This project focuses on correcting phonetic letters in Turkish. With the increase in the use of virtual keyboards on mobile and desktop devices, the Turkish word error rate has increased significantly due to the absence of Turkish letters or typos. Phonetic letter corrections are of great importance to remove this problem in the data that serves many text-based systems and to build a clean database. Because these letters are used extensively in languages such as Romanian, Arabic and Vietnamese, including Turkish, and the presence of these letters also reveals a great difference in terms of meaning. There are similar studies on this subject in Turkish, but one of the most important points of this study is the addition of the letters "â" and "î" to the set of diacritic letters.

In this study, to overcome the mentioned problem, first of all, a raw dataset was prepared with approximately 5000 books on Turkish, both containing a wide variety of words and written in accordance with the spelling rules. This raw dataset was made ready for review and training the system to be created by going through several cleaning steps.

First, N-Gram models were created with these datasets. Zipf and Mandelbrot distributions were observed on the generated models, and perplexity values were calculated to evaluate how well a language model was designed. After the high compatibility values obtained here, these data were used in the next learning and evaluation steps.

In the training and evaluation of the model, which is the last step of the project, letter-based system training was carried out by using the “seq2seq” based learning model. A new artificial input set was created by randomly changing the Latin equivalents of the diacritic letters and phonetic letters in the words on the control sets separated from the main data set. The trained model was evaluated with this control set and success rates over 90% were obtained.

RÉSUMÉ

Ce projet se concentre sur la correction des lettres phonétiques en turc. Avec l'augmentation de l'utilisation des claviers virtuels sur les appareils mobiles et de bureau, le taux d'erreur sur les mots turcs a considérablement augmenté en raison de l'absence de lettres ou de fautes de frappe turques. Les corrections de lettres phonétiques sont d'une grande importance pour éliminer ce problème dans les données qui servent de nombreux systèmes basés sur du texte et pour créer une base de données propre. Parce que ces lettres sont largement utilisées dans des langues telles que le roumain, l'arabe et le vietnamien, y compris le turc, et la présence de ces lettres révèle également une grande différence en termes de sens. Il existe des études similaires sur ce sujet en turc, mais l'un des points les plus importants de cette étude est l'ajout des lettres "â" et "î" à l'ensemble des lettres diacritiques.

Dans cette étude, pour surmonter le problème mentionné, tout d'abord, un ensemble de données brutes a été préparé avec environ 5000 livres sur le turc, contenant tous deux une grande variété de mots et écrits conformément aux règles d'orthographe. Cet ensemble de données brutes a été préparé pour l'examen et la formation du système à créer en passant par plusieurs étapes de nettoyage.
Tout d'abord, des modèles N-Gram ont été créés avec ces ensembles de données. Les distributions Zipf et Mandelbrot ont été observées sur les modèles générés, et les valeurs de perplexité ont été calculées pour évaluer dans quelle mesure un modèle de langage a été conçu. Après les valeurs de compatibilité élevées obtenues ici, ces données ont été utilisées dans les prochaines étapes d'apprentissage et d'évaluation.

Dans la formation et l'évaluation du modèle, qui est la dernière étape du projet, la formation au système basé sur des lettres a été réalisée en utilisant le modèle d'apprentissage basé sur « seq2seq ». Un nouvel ensemble d'entrées artificielles a été créé en changeant au hasard les équivalents latins des lettres diacritiques et des lettres phonétiques dans les mots sur les ensembles de contrôle séparés de l'ensemble de données

principal. Le modèle entraîné a été évalué avec cet ensemble de contrôle et des taux de réussite supérieurs à 90 % ont été obtenus.

ÖZET

Bu proje, Türkçe ’deki fonetik harflerin düzeltilmesine odaklanmaktadır. Mobil ve masaüstü cihazlarda sanal klavye kullanımının artmasıyla birlikte Türkçe harflerin olmaması veya yazım hatası nedeniyle Türkçe kelime hata oranı önemli ölçüde artmıştır. Metin bazlı çalışan birçok sisteme hizmet eden verilerdeki bu sorunu kaldırmak ve temiz bir veri tabanı inşa edebilmek için fonetik harf düzeltmeleri büyük önem arz etmektedir. Çünkü Türkçe de dâhil olmak üzere Rumence, Arapça ve Vietnamca gibi dillerde bu harflerin oldukça yoğun bir kullanımı bulunmakta ve bu harflerin varlığı anlam açısından da büyük farklılık ortaya koymaktadır. Türkçe ’de bu konuya ait benzer çalışmalar mevcut lâkin bu çalışmanın ayrıştığı en önemli noktalardan biri fonetik harfler kümesine “â” ve “î” harflerinin de eklenmesidir.

Bu çalışmada, bahsedilen sorunun üstesinden gelmek için öncelikle Türkçe ‘ye dair hem çok çeşitli kelimeler barındıran hem de yazım kurallarına uygun bir şekilde yazılmış yaklaşık 5000 adet kitap ile ham bir veri kümesi hazırlandı. Bu ham veri kümesi, birkaç temizleme adımından geçirilerek incelemeye ve oluşturulacak sistemi eğitmeye hazır hale getirildi.

İlk olarak bu veri kümeleri ile N-Gram modelleri oluşturuldu. Oluşturulan modeller üzerinde Zipf ve Mandelbrot dağılımları gözlemdi ve ne kadar iyi bir dil modeli tasarlandığını değerlendirmek için “perplexity (karışıklık)” değerleri hesaplandı. Burada elde edilen yüksek uyumluluk değerleri sonrası bu veriler bir sonraki olan öğrenme ve değerlendirme adımlarında kullanıldı.

Projenin son adımı olan, modelin eğitimi ve değerlendirmesinde, “seq2seq” temelli öğrenme modeli kullanılarak, harf bazlı sistem eğitimi yapıldı. Veri kümesinden ayrıştırılan kontrol parçaları üzerindeki kelimelerde fonetik ve fonetik harflerin Latin karşılıkları rastgele olarak değiştirilerek yeni bir yapay girdi kümesi oluşturuldu. Eğitilen

model, oluşturulan bu kontrol setiyle değerlendirildi ve %90 üzerinden başarımlar elde edildi.

1. INTRODUCTION

Social media platform, for example, Twitter have developed at a gigantic speed lately and have turned into a significant wellspring of information giving data incalculable field. The present circumstance was important to analysts and many examinations on AI and regular language preparing was led via social media information. In any case, the language is utilized in online media contains an extremely high measure of boisterous information than the proper composing language diacritic rebuilding is considered as one of the most difficult preprocessing steps in text standardization. Furthermore, technical issues might arise in social media due to text normalization related to character-based noise Thus, diacritic is a set of marks used to change the phonetic properties of letters and is used on many languages besides Turkish.

In big data era, people might prefer shortcuts to reduce the complexity of text typing issues where writing without diacritic letters becomes a standard issue for numerous reasons. Diacritic restoration would be identified as the replacement of the noisy letter with its diacritic counterpart. It only focuses on the correctness of diacritic letters. Its scope is more compact and language oriented compared to the spell correction.

Diacritic letters exist in different languages with different charset in a broad range such as Vietnamese, Arabic, Romanian languages (Grozea, 2012; Do et al., 2013; Azmi & Almajed, 2015). These languages densely use diacritic letters, and the letters are significant in terms of meaning and the pronunciation of a word. This thesis mainly focuses on Turkish language, but it is language independent by its nature. Training with a proper language dataset, it's able to learn and correct diacritic with high accuracy.

Words containing diacritic typos in a certain language would be easily read, corrected, and understood by humans without further assistance, On the other hand, word correction

is a complex problem of text preprocessing since noisy characters do not belong to formal lexicon. Then the text becomes illegible in lexical parsing and even incomprehensible if a proper correction is ignored. Sentiment analysis, speech analysis, text to speech synthesis and many more NLP operations rely on text-based inputs. Most of the inputs of such systems originates from mobile devices and it comes with a problem. The problem is composed of two main steps; an appropriate support for a diacritic language and the user preference to have larger letter buttons on device screens. These two main reasons trigger diacritic typos. To make it apprehensible by computers, diacritic correction takes it over.

Statistical models are widely used in of natural language applications to retrieve language characteristics for multimodal purposes. Natural language analysis requires quantitative tools related to generalized corpus features. In language identification, word-based analysis reflects large scale properties of the language through many computer science applications such as data security, language identification, spell checking, data compression, authorship attribution and speech recognition. In the scope of this study, a large-scale corpus is created and used to discover language characteristics of Turkish. Word and letter-based analyses are made on this corpus to build a base for several NLP studies.

Investigation of statistical properties of a language is useful for cryptanalysis, compression processes, optical character, and speech recognition, etc. For this purpose, a corpus, which is large enough to represent a language, should be created to measure the quantitative parameters and to assure a suitable dataset for the application of diacritic text processing.

Initially, to provide a decent dataset to the project, 5000 books in Turkish are gathered. These books have more 50 million words and 3GB in size. All the books have undergone

editorial checks. So, they are good candidates to create a ground truth. These files contain images, sounds, videos, fonts, and other unnecessary inputs, so just extracted text parts by converting their encodings to UTF-8. By creating adapter processors these texts are converted and merged them into a single file. Only alphanumeric characters are considered, and the rest are ignored, and the cleaning operations are conducted on the single file to produce the project's ultimate dataset called GSUCo.

N-gram models from unigrams to pentagrams are created for GSUCo. On the N-Grams models, their compatibility with some of the de facto rules such as Zipf's Law is inspected. Zipf's Law is a well-known rule to measure the distribution of word in a language model. Another measurement applied to N-Grams is perplexity. These two main rules, gives clue about how accurately composed a language model.

Finally, a sequence-to-sequence model is designed to achieve high accuracy on diacritic corrections. It's trained with GSUCo, and the test sets are also excerpted from GSUCo. Correct letters are randomly replaced with their counterparts to create a synthetic dataset for testing. After training the model, more than %90 accuracy is obtained on the test set.

2. LITERATURE REVIEW

The usage mobile devices raised dramatically in recent years and mobile provides most of the data to the systems that requires large amount of data. Text-based data still has the significant proportion of the data. That's why diacritic correction has gained importance for the languages like Turkish. Diacritic letters affect the meaning of the word, and they may change it completely compared to their Latin equivalent letters. There are previous studies that focuses on both Turkish and other languages containing diacritics.

In Turkish corpus linguistics two major areas are discussed in the literature survey. Corpora analysis and word rankings are measured through statistical laws such as Zipf's and Heap's laws. The first section represents lexical insights of power laws in this domain. The second part is focused on diacritics and current problems of Turkish language in NLP.

2.1 Lexical Analysis and Distributional Laws

Ilgen and Karaoglan (2007) investigated the meaning of Turkish words through Zipfian parameters. They have developed a direct approach to quantify lexical parameters in the Turkish case. Zipfian parameters have been derived from two Turkish corpora on which the meanings are labeled. Their study has contributed to Turkish corpora linguistics in resolving word disambiguation of Turkish lexicons.

Karaoglan et al. (2013) have shown the effects of corpora normalization through statistical parameters where the entropy has been used as a model performance metric. Therefore, they combined Zipf's and Heaps' power laws to represent semantic information on different lengths of sub-corpora by using METU corpora.

In a similar study, Karaoglan et al. (2012) have studied the rank and number of meanings with term-document matrix where latent semantic features were analyzed through Zipfian distributions and Singular Value Decomposition.

Casas et al. (2019) have focused on the meaning-frequency law and the law of abbreviation. They have measured the robustness of the laws on the English language. Moreover, they have extended their study on Dutch and Spanish languages by adding two additional metrics phonemic and syllabic lengths.

Bozsahin (2011) has suggested an extension of semantics to compute distributional features at the level of morphemes. Also, a fusion model has been proposed where morphological and syntactic processing was designed at the level of semantics computation.

2.2 Zipf's Law and Diacritic Restoration studies on Turkish

Dalkılıç and Çebi (2004) observed Zipf's Law compatibility with their dataset called TurCo (Turkish Corpus) in 2004. TurCo is about 360 MB and contains 50M words. They worked on unigram, bigram, trigram, quadgram and pentagram. 0.98 r-squared is observed for unigram, on the other hand quadgram and pentagram did not obey Zipf's Law.

Adalı and Eryiğit (2014) carried out a study on social media data [5]. Vowel and diacritic restorations are aimed and using discriminative sequence classifier and language validator mechanisms, %97.06 word accuracy is achieved as result.

Another project is Özer et al.'s (2018) diacritic restoration of Turkish tweets with word2vec [6]. Corpus contains 2.107.366 tweets and 6.009.229 diacritic words. Restoration is formed by three steps. First generate all possible word with diacritics,

secondly find candidate words from SkipGram model and finally calculate cosine similarities and select the best word. %94.20 accuracy is achieved with this model.

Şahin et al (2013), have analyzed Turkish diacritics through morphology processing. They have developed a two-step morphological analyzer combined with Turkish lexicon and lemmas. Their results have been compared with the pioneering methodology of Oflazer morphological analyzer in order to get the performance for unknown words.

Alpkocak and Ceylan (2012) have investigated the effects of Turkish diacritics on information retrieval. They have focused on five special characters of the Turkish alphabet. They have shown the decrease of retrieval performance with respect to mistyped characters. They have proposed three methods; token normalization, document expansion, and query expansion in order to resolve Turkish diacritics problem in information retrieval.

Küçük and Steinberger (2014) have evaluated opinion mining experiments to recognize named entities for Turkish texts. They have reconsidered mistyped characters with diacritic correction. Even if they have encountered persisting problems through the normalization process, they have carried out named entity recognition by extending a tweet normalization process.

Chetail and Emeline (2019) have taken into account the problem in a different viewpoint. They have processed visual diacritics in human visual perception. They have applied two main experiments and concluded that diacritics contribute to the overall visual shape in vocabulary and diacritic letters are not mere variants of their complementary equivalents.

Klyshinsky et al. (2020) have compared the performance of diacritics restoration using neural networks. They have measured the accuracy metrics for seven languages: Turkish,

Romanian, Slovak, German, French, Croatian and Latvian. They have noted that the performance of diacritic correction is highly correlated with language and provided resources

2.3 Diacritic Restoration studies on other languages

One of most recent study on diacritic restoration in other languages is Romanian Automatic Diacritics Restoration Challenge (Florin et al., 2019). Corpus of the project consists of 40M words and 2.5M sentences. Using two-layer BiLSTM model, character-level learning is applied. With this model %99.46 word-level accuracy reached.
Similarly, the study carried on Vietnamese, used a LSTM model and trained for character-level learning with 150 MB dataset (Hung, 2018). As result %95 word-level accuracy is obtained.

Two of the most popular papers on Arabic diacritization is Multi-components System for Automatic Arabic Diacritization and Automatic diacritization of Arabic text using recurrent neural networks (Abandah et al, 2015; Hamza & Xiong, 2020). In first study, a corpus containing about 75.6 words is utilized. Using LSTM and dense layers, word error rate have been decreased to %6.22. The second project applied Arabic Treebank Part 3 as corpus. Adopting a method called maximum entropy, the word error rate dropped to %7.2.

3. METHODOLOGY

The approach to solve the defined problem is explained in this section. Gathering the dataset, cleaning the dataset and the implementation of analysis are explained in detail. The steps of the solutions in short are:

• The first step is collecting books and categorizing them to have a solid insight. At this step, only books which editorially corrected are selected.

• Data cleaning and the preparation is the second step of the study. Initially, corrupted, and unknown-encoded files are eliminated. Afterwards, the contents of books are merged into a file and then normalized. In normalization, the punctuations and irrelevant characters are removed.

• N-Gram specifications of the corpus are deeply analyzed at this section. The Zipf-Mandelbrot Law's outputs are produced, and the related variables of the law are calculated.

• Finally, by modelling a sequence-to-sequence, diacritic restoration is conducted based on the probabilities of the letters which are fed by token of n-grams. There are three different train and test sets in respect %90 - %10, %80 - %20 and %70 - %30.

The overall flow of the project is shown in Figure 1.

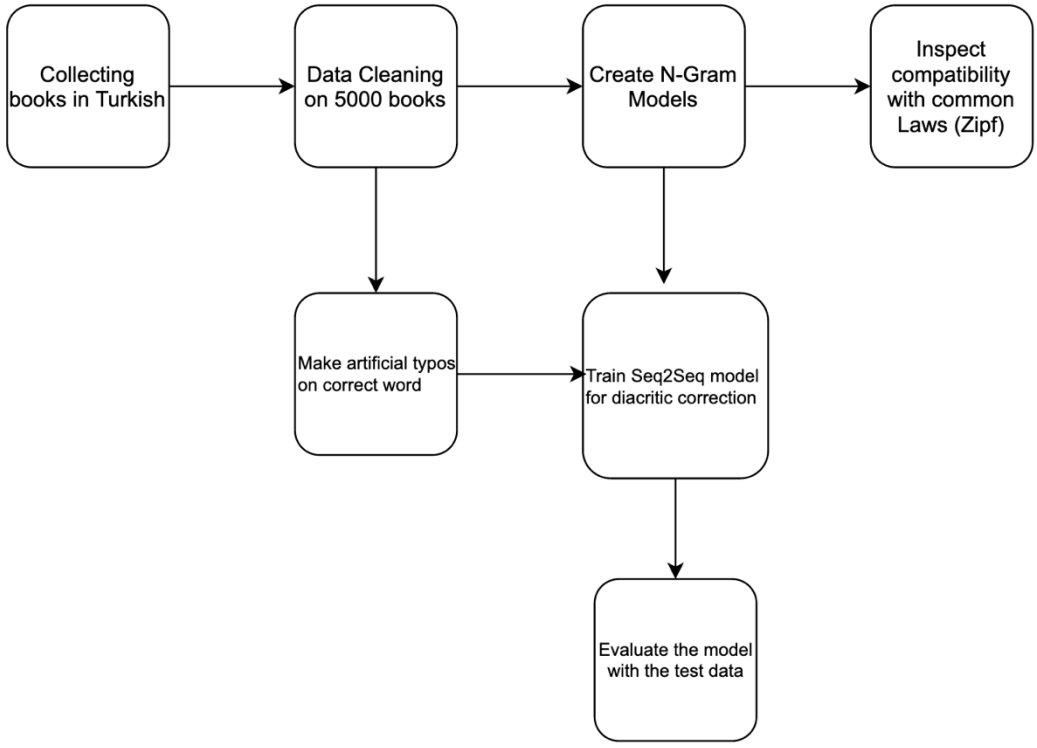

Figure 1: General flow of the project

3.1 Dataset

Since it is crucial to have a clean corpus, Turkish books are selected as they are undergone editorial checks. Dataset of the project, called GSUCo, consists of 4975 Turkish books which are encoded in UTF-8. Two types of adapters are written convert EPUB and PDF files into text files in Python. After merging books into a single file, cleaning operation

conducted such removing apostrophes or any other punctuations. The books have more than 330 millions of words and 17 millions of sentences in total and the size makes up to 3GB.

It is crucial to have extensive dataset to cover various states of words since words with diacritic marks are common in Turkish. Therefore, to have unbiased and comprehensive dataset, books in different categories are selected. The distribution of the books is shown in Table 1.

Book Category	*Percentage*
Novel	31,60%
History	8,80%
Art	2,40%
Poetry	2,00%
Politics & Diplomacy	8,40%
Philosophy	8,00%
Biograph	1,20%
Other	8,80%
Economy & Work	2,00%
Religious	6,80%
Sci-fi & Fantasy	6,80%
Anagraph	1,20%
Story & Tale	7,20%
Detective	4,80%

Table 1: Books by category used for corpus

Since the diacritics are the focus of the project, the distribution of the diacritic letters is vital. That's why to have better insight about the dataset, the distribution is calculated, and the results can be seen in Table 2.

Letter	Count
A	234.165.045
Â	4.297.604
C	19.519.414
Ç	22.030.076
G	26.237.366
Ğ	21.460.627
I	98.473.144
İ	177.711.812
Î	1.500.549
O	52.858.844
Ö	16.309.614
S	64.401.308
Ş	34.374.279
U	67.296.656
Ü	37.687.034
Total Letter Count	2.067.705.795

Table 2: Distribution of Diacritic Letter in Dataset

3.1.2 Entropy & Perplexity

Statistics are estimated to extract the information content of data. An N-gram language model can be viewed as a statistical model trying to predict the next word given N-1 previous words. This approach is borrowed from information theory, which was constructed by Shannon. The information content of a random variable X is measured by the concept of entropy (Shannon, 1948). Entropy can be viewed as the average uncertainty about occurrence of an event. It is formulated as follows:

$$H(X) = -E[\log P\{X\}] \quad 1$$

So, when the probability distribution of X with finite alphabet of size L is $\{p_0, p_1, \dots p_{L-1}\}$ then entropy can be written as

$$H(p_0, p_1, \dots p_{L-1}) = -\sum_{0}^{L-1} p_i \log p_i \quad 2$$

The main responsibility of a statistical language model is to estimate the "true" probability distribution of the language, P{X}. If the distribution is misestimated then the entropy, in other words the uncertainty about the next event will be higher. This entropy is called the "cross entropy", which includes the true uncertainty plus the uncertainty added by the wrong estimation of probability distribution:

$$cross - entropy(P; P_{LM}) = -\sum_{x} P\{X\} \log P_{LM}\{X\} \quad 3$$

where $P_{LM}\{X\}$ is the estimated probability distribution of a particular language model. Since P$\{X\}$ is unknown, another measure should be utilized to find the cross entropy. That measure is called "logprob (LP)" and is defined as

$$LP = \lim_{n\to\infty} -\frac{1}{n}\sum_{k=1}^{n} \log P_{LM}(w_k/w_1, .., w_{k-1}) \quad 4$$

where $\sum_{k=1}^{n} \log P_{LM}(w_k/w_1, .., w_{k-1})$ is the log probability of the long sequence $w_1 .., w_n$ estimated by the language model. So instead of the ensemble average, time average is taken to estimate the cross entropy with an assumption that language is stationary and ergodic. Since language is neither stationary nor ergodic the logprob is an approximation to the cross entropy.

Logprob is estimated on a sequence of unprecedented text, called test text. If this text appears in the training text, the chances are high; consequently, the cross entropy will be estimated to be less than it should be. The cross entropy is always greater than the entropy itself, and the quality of the language model is determined by how closely the cross entropy approaches the language entropy. Entropy is also defined as the lower limit on the number of bits or the number of "yes/no" questions to encode or predict further information. Therefore, the average number of choices can be formulated as

$$PP = 2^{LP} \quad 5$$

where the quantity PP is called “perplexity”. From the identifier's point of view, perplexity gives the average number of subsequent word choices to predict. When perplexity is high, the identification task becomes more difficult due to the large number of choices. This difficulty arises from the language itself (language entropy) and the language model (additional uncertainty when statistics are incorrectly estimated). Based on the n-gram calculation conducted on the dataset, the perplexity and entropy results are shown in Table 3

	Unigram	Bigram	Trigram
Entropy	8.95	7.91	6.86
Perplexity	495.61	241.71	116.49

Table 3: Entropy & Perplexity Results

3.2 N-Grams

Markov chains calculates the probability of transition to future states for given conditions. N-gram is a specialized for of Markov chains. N-Gram is basically sequence of word and N indicates the number of words in the gram. It is used to predict next word for any given N-1 words. By processing the given dataset, the model creates a cloud of probabilities of words. By doing so, the most likelihood word for a given sequence is predicted. In the project, we inspected the distributions of unigram, bigram, trigram, quadgram and pentagram. The number of tokens in each gram analysis is shown in Table 1. Formulas for unigram, bigram and trigram are shown respectively below;

$$P(word_i) = \frac{C(word_i)}{\sum_{k=0}^{n} C(word_k)} \quad (6)$$

$$P(word_i \mid word_{i-1}) = \frac{C(word_{i-1}, word_i)}{C(word_{i-1})} \quad (7)$$

$$P(word_i \mid word_{i-1}, word_{i-2}) = \frac{C(word_{i-2}, word_{i-1}, word_i)}{C(word_{i-2}, word_{i-1})} \quad (8)$$

Where $C(word_i)$ is the count of $word_i$ in the corpus.

3.3 Zipf's Law

Zipf(1949) discovered the law which states that for a given natural language corpus, frequencies of words are inversely correlated to their rank. According to this law, if f is the frequency of a word in the corpus and r is the rank, then:

$$f = \frac{k}{r} \tag{9}$$

Zipf's law generates a linear line with almost a slope -1. After the discovery of the law, Mandelbrot brought forward a generalized version of Zipf's law to provide better fitting in low-rank area. The new formula is:

$$f = \frac{k}{(r+c)^d} \tag{10}$$

where c and d are constants belonging to the corpus and most of the time, c and d are found to be small deviations compared to Zipf's law (1).

The Levenberg-Marquardt (LM) algorithm is an iterative technique that locates the minimum of a multivariate function expressed as the sum of the squares of nonlinear real-valued functions. It has become a standard technique for nonlinear least squares problems that is widely used in a wide range of disciplines. LM can be seen as a combination of the steepest descent and the Gauss-Newton method. When the current solution is far from the correct one, the algorithm behaves like a steepest descent method: slow but guaranteed to converge. To figure out the optimal values for Mandelbrot equation, at first, we moved the eq 2 on log base to get the correlation between variables easier. Least Squares Error, initial coefficients of the new equation were found via by applying the Nonlinear Least Squares method and these coefficients were placed to find the parameters of original equation using Levenberg-Marquardt function.

$$\log f = \log k - d.\log(r+c) \tag{11}$$

3.4 Neural Network

Neural network is a computer system which is similar to human brain and also a technique for gaining experience from data. The system is made of multiple perceptron in each layer and the outputs transferred to next layer until final value step reached. Each perceptron has its own activation function to create an output. The simple structure of neural network can be seen in Figure 2.

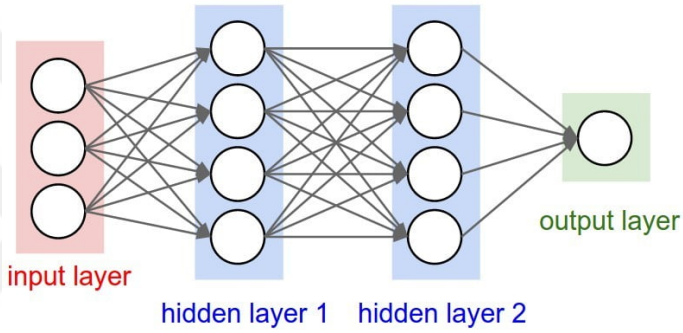

Figure 2: Structure of a Neural Network

3.5 RNN Architecture for Diacritic Restoration

A recurrent neural organization (RNN) is a type of neural organization that uses sequential information or time series information. These deep learning calculations are typically used for mundane or mundane things, such as language interpretation, natural language processing (NLP), speech recognition, and image inscription; bundled into popular apps like Siri, voice search, and Google Translate. Like feedforward and convolutional neural organization (CNN), iterative neural networks use information preparation for learning. They are recognized by their "memory" when they retrieve data from previous contributions to influence current information and performance. While an ordinary deep neural network accepts that information sources and rendering are independent of each other, recurrent neural network rendition builds on previous components in the cluster. While future probabilities would be equally useful in

determining the outcome of a given setting, unidirectional recurrent neural networks were unable to account for these probabilities in their predictions.

3.6 Seq2Seq Model

The final step of the project is implementation an encoder-decoder LSTM architecture for seq2seq diacritic correction. LSTM is a specialized version of RNN which handles vanishing gradient problem. Internal architecture is shown in Figure 3 (Yuan et al., 2021).

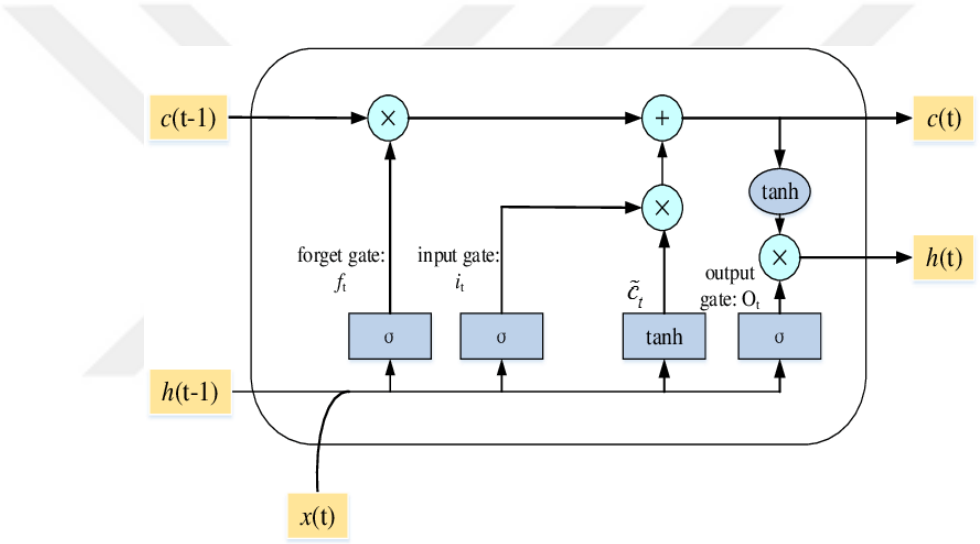

Figure 3: The structure of the LSTM unit

tanh is used as the activation function. seq2seq is a general-purpose encoder- decoder framework that can be used for Machine Translation, Text Summarization, Conversational Modeling, Image Captioning, and more. That's for our diacritic correction aim, seq2seq model is preferred. It is simply based two LSTM models called encoder and decoder.

Encoder peruses the info arrangement and sums up the data in something many refer to as the inside state vectors or setting vector (if there should arise an occurrence of LSTM these are known as the hidden state and cell state vectors). It is disposed of the yields of

the encoder and just safeguard the inside states. This setting vector intends to epitomize the data for all information components to help the decoder make precise expectations.

The decoder is an LSTM whose initial state is initialized to the final state of the LSTM encoder, i.e. the context vector of the encoder's last cell is inserted into the first cell of the decoder network. Using this initial state, the decoder starts generating a sequence of outputs and this output is also accounted for subsequent outputs. The basic structure of a seq2seq model is shown in Figure 4 (Shukla et al., 2021). It simply gets input characters and converts them into the desired format. It can be translation or word correction.

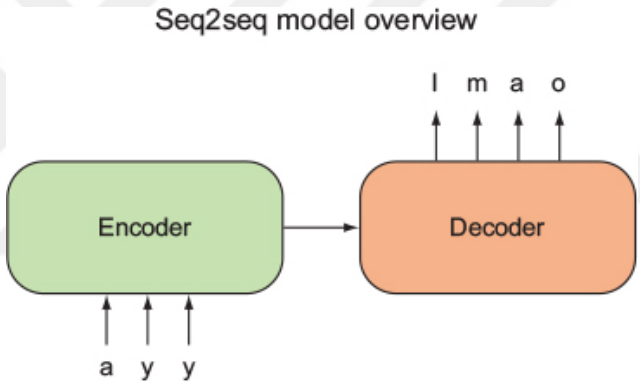

Figure 4: Simple structure of seq2seq model

The character-level diacritic seq2seq checker is trained on unigram tokens derived from a vocabulary of 30k unique Turkish words.

At the evaluation step, two main metrics are inspected which are accuracy and diacritic correctness. For accuracy metric, a word is accepted as a single unit and its full match is counted as accuracy. In diacritic correctness, the equality of each diacritic letter in a word is observed.

4. RESULTS

Firstly, N-Gram models are created and observed on GSUCo. Table 4 shows the numbers of total tokens for each N-Gram model.

Gram	Number of Tokens
Unigram	3.399.137
Bigram	87.589.716
Trigram	202.743.188
Quadgram	243.400.999
Pentagram	246.466.177

Table 4: Count of sequences in N-Grams

The 10 most common words in grams can be seen in Table 5.

Unigram		Bigram		Trigram	
Frequency	Token	Frequency	Token	Frequency	Token
10.219.598	bir	465.149	ya da	53.981	başka bir şey
7.299.319	ve	396.539	bir şey	36.718	bir kez daha
4.264.130	bu	188.499	büyük bir	32.832	bir süre sonra
2.604.511	de	174.750	başka bir	32.634	ne var ki
2.526.809	da	166.002	ne kadar	24.305	ama yine de
1.948.679	için	158.233	ve bu	22.408	kısa bir süre
1.615.454	o	139.764	bir şekilde	20.049	bir yandan da
1.553.581	gibi	138.333	o kadar	18.366	ne olursa olsun
1.513.064	daha	135.922	diye sordu	18.057	ya da bir
1.474.025	ama	128.896	o zaman	17.567	her ne kadar

Table 5: Ten most common tokens in unigram, bigram and trigram

As result, after completing first model which is N-Gram, the distributions of the grams are analyzed and checked if it complies with Zipf rule and smoothed with Mandelbrot constants. Figure 4 shows the accordance between language model and Zipf law with 0.95 r-squared value. The closer the r-squared to 1, the better the language model is. After applying Mandelbrot analysis to the curve in Figure 5, an improved result is produced shown in Figure 6, with 0.988 r-squared value which a sign of well-designed language model.

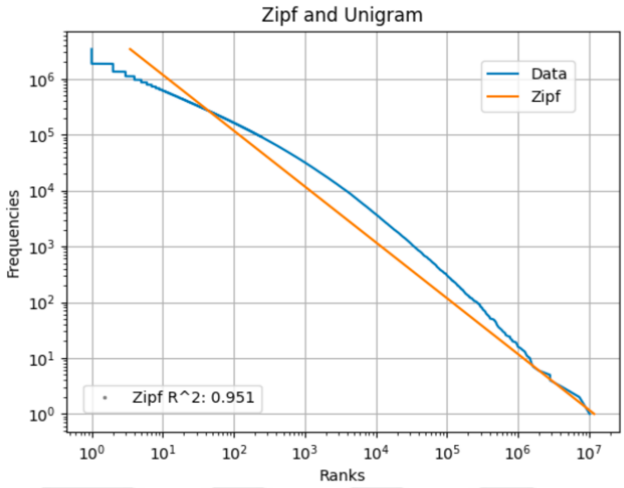

Figure 5: Unigram Zipf Curve for the GSUCo

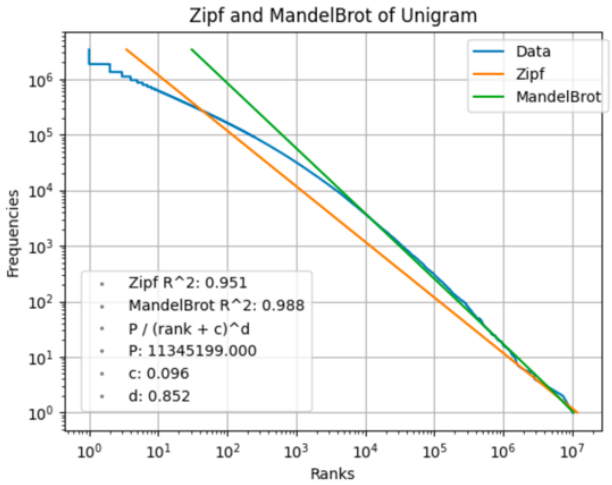

Figure 6: Unigram Zipf and Mandelbrot curves for the GSUCo

We also get good fitting values for Zipf of bigram and trigrams, but it started to decrease after quadgram. However, MandelBrot has made it better for quadgram, pentagram has

about 0.71 fitting value. Figure 7, 8, 9 and 10 accordance Zipf and Mandelbrot of bigram, trigram, quadgram and pentagram induvially and respectively.

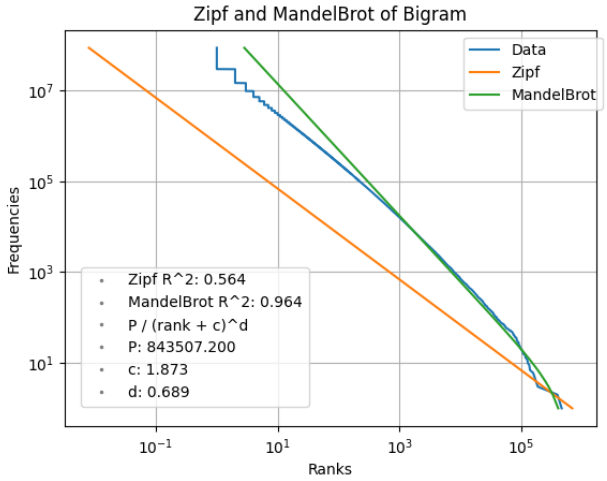

Figure 7: Bigram Zipf and Mandelbrot curves for the GSUCo

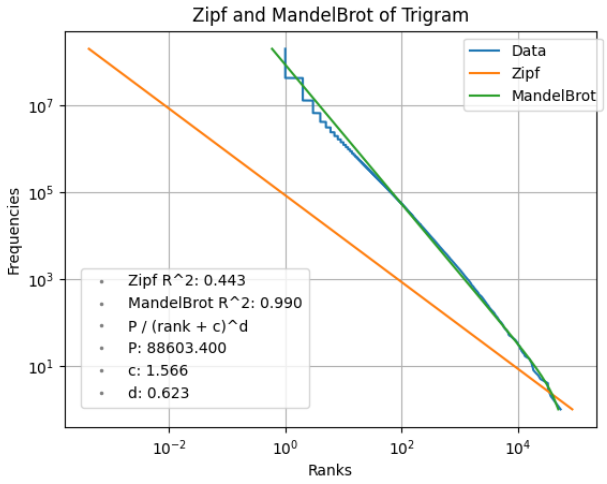

Figure 8: Trigram Zipf and Mandelbrot curves for the GSUCo

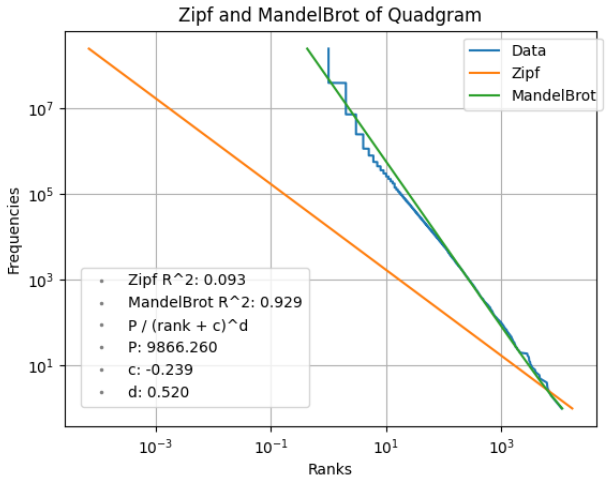

Figure 9: Quadgram Zipf and Mandelbrot curves for the GSUCo

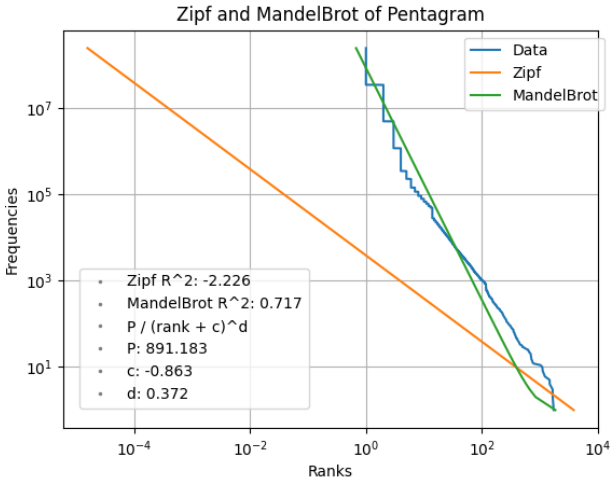

Figure 10: Pentagram Zipf and Mandelbrot curves for the GSUCo

Figure 11 shows the distribution of Zipf for all N-Grams created of this study. Frequency rank and absolute frequency of word is shown in axes x and y, respectively.

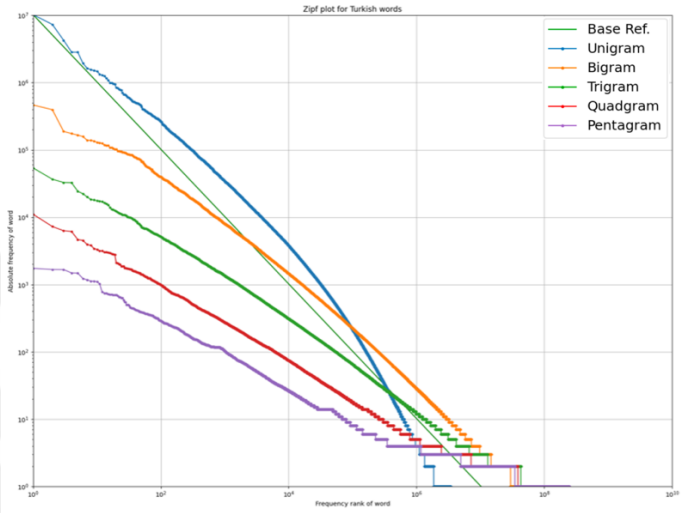

Figure 11: Zipf Distribution

The fitting values for Zipf and Mandelbrot are shown in Figure 12. It is observed that although they have higher fits in unigram, bigram and trigram, R-Square values are dropped dramatically in quadgram and pentagrams and Figure 12 shows this behavior. The r-squared values of Zipf curves closer to the 1 up to quadgram (unigram 0.95, bigram 0.56, trigram 0.44, quadgram 0.09), however pentagram's -2.26. After finding out Mandelbrot constants for each n-gram, the r-squared values are increased, 0.98, 0.96, 0.99, 0.92, 0.71 for unigram to pentagram, respectively.

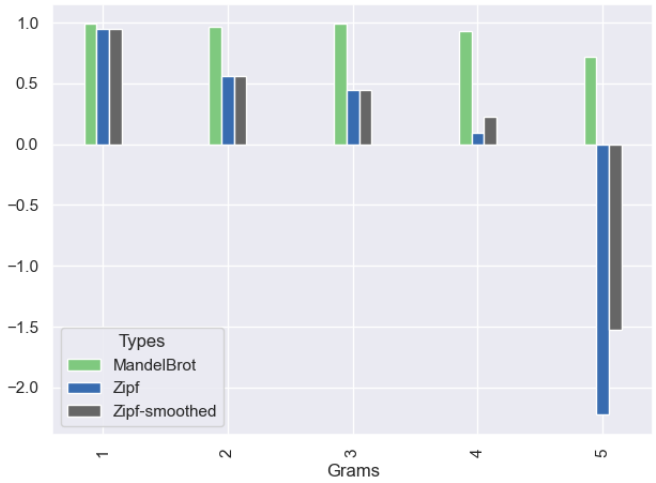

Figure 12: R-Square values for Zipf and Mandelbrot

With the help of the N-Gram models, a word recommender system is designed. This system tries to figure out the next and the most likelihood word in a sentence while typing. For limited training size, it could reach %67.41 accuracy for the predictions.

After modeling and creating N-Gram models, a LSTM model is designed to predict correct letter for a corrupted word in test. The test set are prepared by randomly assigning the opposite letter for diacritics such “c” for “ç”, “ö” for “o” and so on.

For the evaluation of the LSTM model, a sub-dataset with 20.000 sentences is created. Evaluation consists of three inputs correct word, test (input) word and estimated word. For each word in test set, a predicted word is created. Then to measure the correctness of the model, diacritic letters in correct and predicted words are compared. Equality in this comparison results in diacritic correctness. The full match of the words creates word correctness. The LSTM model was evaluated with this method and the results are shown in Table 6.

Letter	Correctness (%)
A	98.25
Â	38.34
C	93.35
Ç	97.53
G	97.59
Ğ	88.06
I	97.63
İ	89.41
Î	48.44
O	94.61
Ö	96.39
S	95.98
Ş	88.18
U	94.38
Ü	95.44

Table 6: Correctness of predicted diacritic letters

In overall, %94.51 of diacritic letters are truly predict via the model. In the means of words correctly predicted, the accuracy is %87.19. As an example, once the input given to the model like “öğrencıların tumü şınifı başarıyla geçmıstı.”, it can successfully transform it into “öğrencilerin tümü sınıfı başarıyla geçmişti”. As it can be seen in the example, the model takes the diactiric changes into account bidirectionally unlike some previous studies.

There are two remarkable lower correctness values belong to “â” and “î” compared to others. For the letter “â” there are different words that have both “â” and “a” like, “kar” (snow) and “kâr” (“profit”) or “hala” (aunt) and “hâlâ” (still). Since the model relies on character-level probabilities these words cause ambiguity for it. The same reason is valid

for “î”. To overcome this problem more samples with the letters should be added to corpus.

5. CONCLUSION

Diacritic correction occupies an important place for text-based learning models. Since typos of diacritic letters changes meanings of words completely, it is crucial to have correct and clean dataset to train a model with it. As Turkish is an agglutinative language, words may have multiple forms to express the state. The number of changes due to rules and structure of the language, entails a large and clean corpus.

In this study, to satisfy the requirements for a solid learning model, approximately 5000 books have been used and its size of is larger than most of the previous studies. Firstly, in the preprocess stage, these books are prepared to be both monitor language model features and to be given as input to the learning model. Afterwards, N-Gram modelling and inspecting the compatibility with common laws have given results as expected. Up to trigrams, the fitness of the distributions of tokens in n-grams are acceptable, however quadgram and pentagram tokens could not perform as smaller ones.

In the phase of learning, LSTM based Seq2Seq model was preferred because its high success rate on sequential inputs such as natural languages. About %94 of diacritic accuracy is achieved by doing so. There are two main drawbacks using such systems, the size of the corpus and the computational power. If power is renounced, the required time raises dramatically. Therefore, the dataset could not be utilized completely to train the model. Although this model could not outperform all antecedent projects, it’s the first to take the long vowels (â and î) into account. That’s why, we believe that with more computing power and using the corpus fully will result in higher accuracy values.

REFERENCES

Abandah, G., Graves, A., Al-Shagoor, B., Arabiyat, A., Jamour, F., & Al-Taee, M. (2015). Automatic diacritization of Arabic text using recurrent neural networks. *International Journal On Document Analysis And Recognition (IJDAR)*, 18(2), 183-197.

Abbad, H., & Xiong, S. (2020). Multi-components System for Automatic Arabic Diacritization. *Advances in Information Retrieval 12035,* 341 - 355.

Adali, K., & Eryiğit, G. (2014). Vowel and diacritic restoration for social media texts, in *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*.

Alpkocak, A., & Ceylan M. (2012). "Effects of diacritics on Turkish information retrieval." *Turkish Journal of Electrical Engineering & Computer Sciences* 20.5 787-804.

Azmi, A.M., & Almajed R.S. (2015). A survey of automatic Arabic diacritization techniques, Natural Lang. Eng. 21 (3), 477–495.

Bozsahin, C. (2011). Morphological preprocessing or parsing: where does semantics meet computation?, *Foundation of the national speech and language technologies platform workshop on the current status of research on turkish*.

Casas, B., et al. (2019). Polysemy and brevity versus frequency in language, *Computer Speech & Language* 58, pp. 19-50.

Chetail, F., & Boursain E. (2019) Shared or separated representations for letters with diacritics?, *Psychonomic bulletin & review* 26.1, 347-352.

Dalkiliç, G., & Çebi, Y. (2004). Zipf's Law and Mandelbrot's Constants for Turkish Language Using Turkish Corpus (TurCo). *ADVIS*.

Do, T.N.D., Nguyen D.B., Mac D.K.& Tran, D.D. (2013). Machine translation approach for vietnamese diacritic restoration, *in: Asian Language Processing (IALP)*, 2013 International Conference, IEEE, pp. 103–106.

Florin, I., Lucian G., Dan O., & Horia C. (2019). Romanian automatic diacritics restoration challenge, in *Proceedings of the 14th International Conference Linguistic Resources and Tools for Natural Language Processing*, pp. 65–74.

Grozea C. (2012). Experiments and results with diacritics restoration in Romanian, *in: International Conference on Text, Speech and Dialogue*, Springer, Berlin, Heidelberg, pp. 199–206.

Hung, B. T. (2018). Vietnamese Diacritics Restoration Using Deep Learning Approach, *in Proceedings of the 10th International Conference on Knowledge and Systems Engineering – KSE*

Ilgen, B., & Karaoglan, B. (2007). Investigation of Zipf's 'law-of-meaning'on Turkish corpora. *2007 22nd international symposium on computer and information sciences*. IEEE.

Karaoĝlan, B., Aslan, S. & Karayer, E. (2012) The Zipfian distributions on term x document matrix. *AIP Conference Proceedings*. Vol. 1470. No. 1. American Institute of Physics.

Karaoğlan, B., et al. (2013). A proposal for corpus normalization., *2013 21st Signal Processing and Communications Applications Conference (SIU)*. IEEE.

Klyshinsky, E., Olesya K., & Alexander B. (2020). A Comparison of Neural Networks Architectures for Diacritics Restoration., *International Conference on Analysis of Images, Social Networks and Texts*. Springer, Cham,.

Küçük, D. & Steinberger, R. (2014). Experiments to improve named entity recognition on Turkish tweets. arXiv preprint arXiv:1410.8668

Ozer Z., Ozer, I. & Findik, O. (2018). Diacritic restoration of Turkish tweets with word2vec, *Engineering Science and Technology, an International Journal*, vol. 21, no. 6, pp. 1120–1127.

Sahin, M., Sulubacak U. & Eryigit G. (2013). "Redefinition of Turkish morphology using flag diacritics." *Proceedings of The Tenth Symposium on Natural Language Processing (SNLP-2013), Phuket, Thailand, October*.

Shukla, N., et al. (2018). Chapter 11. Sequence-to-Sequence Models for Chatbots, *Machine Learning with Tensorflow*, Manning, Shelter Island, NY.

Yuan, X., Li, L., & Wang, Y. (2020). Nonlinear Dynamic Soft Sensor Modeling With Supervised Long Short-Term Memory Network. *IEEE Transactions on Industrial Informatics*, 16, 3168-3176.

BIOGRAPHICAL SKETCH

Hüseyin Ekici has B.S. degree in Computer Engineering from Istanbul Technical University. After graduating in 2015, he has started professional career as a software developer. In 2018, he has begun master's studies in Galatasaray University and been working on machine learning and natural language processing till present.