

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL

**SYLLABLE-BASED APPROACH WITH DEEP LEARNING
METHODS ON TURKISH DIACRITIC RESTORATION**

ÖZNUR ACAR

**A THESIS OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER ENGINEERING**

ADVISOR: PROF DR. YUSUF SİNAN AKGÜL

JULY 2024

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL

**SYLLABLE-BASED APPROACH WITH DEEP
LEARNING METHODS ON TURKISH DIACRITIC
RESTORATION**

ÖZNUR ACAR

**A THESIS OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER ENGINEERING**

ADVISOR: PROF DR. YUSUF SİNAN AKGÜL

JULY 2024

**T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

**TÜRKÇE DİAKRİTİK RESTORASYONU KONUSUNDA
DERİN ÖĞRENME YÖNTEMLERİYLE HECE
TABANLI YAKLAŞIM**

ÖZNUR ACAR

**YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

DANIŞMAN: PROF. DR. YUSUF SİNAN AKGÜL

TEMMUZ 2024

MASTER of SCIENCE JURY APPROVAL FORM

A thesis submitted by Öznur ACAR defended on 25/07/2024 before the jury formed with the 08/07/2024 date and 2024/35 numbered decision of the GTU Graduate Administration Board, has been accepted as a MASTER of SCIENCE thesis in the Department of Computer Engineering.

JURY

MEMBER

(THESIS ADVISOR) : Prof. Dr. Yusuf Sinan AKGÜL

MEMBER : Assist. Prof. Burcu YILMAZ

MEMBER : Assoc. Prof. Hidayet TAKÇI

APPROVAL

Gebze Technical University Graduate Administration Board
...../...../..... date and/..... numbered decision.

SIGNATURE/SEAL

*To my Family
and*

*To my
Husband*

ABSTRACT

The amount of digital text available in multiple languages is constantly expanding due to the widespread usage of the Internet and the quick advancement of technology. However many diacritical marks are absent or misused due to variations in keyboard layouts and alphabets, which can make reading difficult. Applications requiring natural language processing to accurately interpret word meanings in spite of these flaws are challenged by this. The present investigation centers on diacritic restoration (DR), an essential component of numerous natural language processing (NLP) applications spanning various languages.

Given Turkish's strong syllable sensitivity in meaning, a syllable-based bidirectional transformer structure has been developed in this work. To further enhance the model's performance, the syllable, and word structure restrictions and rules were successively coded on the training set. By introducing these structural restrictions and rules into the data during model training, the model's ability to learn semantics was particularly enhanced.

As a result, the study's conclusions emphasize the notable advancement made possible by refining the configuration of the suggested model, surpassing the results of previous word-based and character-based research. We achieved a remarkable 99.02% accuracy rate. Specifically, the extremely high accuracy rate of 97.06% in resolving ambiguous words indicated successful semantic learning. This accomplishment highlights the potential for enhancing NLP applications across a range of languages and represents a significant milestone in the field of diacritic restoration.

Keywords: Natural Language Processing, Diacritics Restoration, Accent Correction, Sequence-to-Sequence Learning, Transformer Model, Syllable-Based Data

ÖZET

İnternetin yaygınlaşması ve teknolojinin hızla ilerlemesi nedeniyle birden fazla dilde mevcut olan dijital metinlerin miktarı sürekli artmaktadır. Ancak klavye düzenleri ve alfabelerdeki farklılıklar nedeniyle birçok aksan işareti yok veya yanlış kullanılıyor, bu da okumayı zorlaştırabiliyor. Bu kusurlara rağmen kelime anlamlarını doğru bir şekilde yorumlamak için doğal dil işlemeyi gerektiren uygulamalar bu konuda zorluk yaşamaktadır. Mevcut araştırma, çeşitli dilleri kapsayan çok sayıda doğal dil işleme (NLP) uygulamasının temel bir bileşeni olan aksan restorasyonuna (DR) odaklanmaktadır.

Türkçenin anlam açısından güçlü hece duyarlılığı dikkate alınarak bu çalışmada hece temelli çift yönlü dönüştürücü yapı geliştirilmiştir. Modelin performansını daha da arttırmak için hece ve kelime yapısı kısıtlamaları ve kuralları eğitim seti üzerinde başarılı bir şekilde kodlanmıştır. Model eğitimi sırasında verilere bu yapısal kısıtlamaların ve kuralların eklenmesiyle, modelin anlambilimi öğrenme yeteneği özellikle geliştirildi.

Sonuç olarak, çalışmanın sonuçları, önerilen modelin konfigürasyonunun iyileştirilmesiyle mümkün olan ve daha önceki kelime tabanlı ve karakter tabanlı araştırmaların sonuçlarını geride bırakan dikkate değer ilerlemeyi vurgulamaktadır. %99,02 gibi dikkat çekici bir doğruluk oranına ulaştık. Spesifik olarak, belirsiz kelimeleri çözmedeki %97,06'lık son derece yüksek doğruluk oranı, başarılı anlamsal öğrenmenin göstergesiydi. Bu başarı, çeşitli dillerde NLP uygulamalarını geliştirme potansiyelini vurguluyor ve aksan restorasyonu alanında önemli bir kilometre taşı temsil ediyor.

Anahtar Kelimeler: Doğal Dil İşleme, Diakritik Restorasyon, Aksan Düzeltmesi, Sekanstan Sekansa Öğrenme, Transformer Model, Hece Tabanlı Veri

ACKNOWLEDGEMENTS

I thank my supervisor Prof. Dr. Yusuf Sinan AKGÜL for giving me a chance to experience in NLP subject and always supporting me. He always lighted my way during my studying.

To my family, my father Kadri BOZDEMİR, my mother Muhterem Emine BOZDEMİR, my sister Özlem ŞEKER and my brother Mehmet Ali BOZDEMİR, thank you for your continuous encouragement and support during my academic journey. My biggest source of support and inspiration has been your unwavering faith in me. Through every challenge and triumph, you have always been there to support and encourage me.

To my husband, Çağlar ACAR, thank you for your love, patience, and understanding. This journey would not have been possible without your steadfast support and sacrifices. You supported me during the difficult moments and long nights, giving me the courage and will to carry on. I am deeply grateful for your companionship and encouragement.

Your support has been instrumental in my success, and I am forever thankful.

TABLE OF CONTENT

	<u>Page</u>
ABSTRACT	vi
ÖZET	vii
ACKNOWLEDGEMENTS	viii
TABLE OF CONTENT	ix
LIST OF SYMBOLS AND ABBREVIATIONS	xi
LIST OF FIGURES	xii
LIST OF TABLES	xiii
1. INTRODUCTION	1
1.1. Diacritic Restoration	4
1.1.1. Diacritic Restoration in Turkish	6
1.1.2. Diacritic Restoration in Other Languages	7
1.2. Training Models	8
1.2.1. Statistical Models	8
1.2.1.1. N-gram Models	10
1.2.1.2. Hidden Markov Models (HMMs)	12
1.2.1.3. Naive Bayes Classifier	15
1.2.1.4. Maximum Entropy Models (MaxEnt)	17
1.2.1.5. Conditional Random Fields (CRFs)	18
1.2.1.6. Latent Dirichlet Allocation (LDA)	19
1.2.2. Deep Learning Models	22
1.2.2.1. Recurrent Neural Networks (RNNs) and Variants	22
1.2.2.1.1. Long Short-Term Memory Networks (LSTM)	24
1.2.2.1.2. Bidirectional Long-Short Term Memory (BiLSTM)	27
1.2.2.1.3. Gated Recurrent Units (GRUs)	29
1.2.2.2. Convolutional Neural Networks (CNNs)	30
1.2.2.3. Sequence-to-Sequence (Seq2Seq) Models	33
1.2.2.4. Transformer Models	35
1.2.2.4.1. Attention is All You Need (Original Transformer)	35
1.2.2.4.2. Bidirectional Encoder Representations Transformer (BERT)	38
1.2.2.4.3. Generative Pre-trained Transformer (GPT)	39
1.2.2.4.4. T5 (Text-to-Text Transfer Transformer)	42
1.2.2.4.5. RoBERTa (Robustly optimized BERT approach)	43

1.3.	Common Approaches for Diacritic Restoration	44
1.3.1.	Word-Based Approach	44
1.3.2.	Character-Based Approach	45
1.3.3.	Syllabel-Based Approach	46
1.4.	Evaluation Metrics	46
1.4.1.	Accuracy	46
1.4.2.	Precision	47
1.4.3.	Recall	47
1.4.4.	F1 Score	47
2.	PURPOSE OF THE STUDY	49
2.1.	Syllable-Based Approach for Turkish Diacritic Restoration	50
2.2.	Sequence to Sequence Transformer Model	51
3.	EXPERIMENT	53
3.1.	Model	53
3.2.	Corpus	55
3.3.	Data Preparation	56
3.3.1.	Removing Punctuation Marks	56
3.3.2.	Replacing Symbols with Their Word Equivalents	56
3.3.3.	Use Only Lower-Case Characters	57
3.3.4.	Representation of Numbers with Common Symbols	57
3.4.	Experiments	58
3.4.1.	Experiment-1: Syllable-Based Training with Semantic Makers	58
3.4.2.	Experiment-2: Total Number of Syllables Indexed Labeling	60
3.4.3.	Experiment-3: The Syllables Indexed Sequential Labeling	61
3.4.4.	Experiment-4: The State of Art Model Syllable Sequential Labeling	62
4.	RESULTS	66
4.1.	Overall Experiments Syllable-Based Results	67
4.2.	Syllable Sequential Labeling Results	67
4.2.1.	Character-Based Result	67
4.2.2.	Word-Based Result	69
4.2.3.	Ambiguous Words-Based Result	69
5.	CONCLUSIONS	71
	REFERENCES	73
	BIOGRAPHY	80
	PUBLICATIONS AND PRESENTATIONS FROM THE THESIS	81

LIST OF SYMBOLS AND ABBREVIATIONS

GTU	: Gebze Technical University
NLP	: Natural Language Processing
DR	: Diacritic Restoration
ADR	: Automatic Diacritic Restoration
DL	: Deep Learning
OOV	: Out of Vocabulary
Seq2Seq	: Sequence to Sequence
HMM	: Hidden Markov Model
MaxEnt	: Maximum Entropy Model
MEMMs	: Maximum Entropy Markov Models
CRF	: Conditional Random Fields
LDA	: Latent Dirichlet Allocation
DER	: Diacritic Error Rate
LM	: Language Model
MLM	: Masked LM
PP	: Perplexity
DNNs	: Deep Neural Networks
FFNN	: Feed-Forward Neural Network
MLP	: Multi-Layer Perceptron
MLE	: Maximum Likelihood Estimation
NSP	: Next Sentence Prediction
FSTs	: Finite State Transducers
RNN	: Recurrent Neural Network
LSTM	: Long Short-Term Memory Network
BiLSTM	: Bidirectional Long-Short Term Memory
GRU	: Gated Recurrent Units
CNN	: Convolutional Neural Network
GPT	: Generative Pre-trained Transformer
T5	: Text-to-Text Transfer Transformer
BERT	: Bidirectional Encoder Representations from Transformer
Syl-B	: Syllable-Based
Char-B	: Character-Based
Exp1	: Experiment 1
Exp2	: Experiment 2
Exp3	: Experiment 3
Exp4	: Experiment 4

LIST OF FIGURES

	<u>Page</u>
Figure 1.1: An example of weather prediction using Markov Chains	13
Figure 1.2: LDA Probabilistic Model	21
Figure 1.3: Recurrent Neural Networks Language Model	23
Figure 1.4: LSTM Cell	25
Figure 1.5: LSTM Architecture	26
Figure 1.6: BiLSTM Architecture	27
Figure 1.7: GRU Cell	29
Figure 1.8: CNN Architecture	31
Figure 1.9: A CNN architecture sample used for image classification	32
Figure 1.10: Seq2Seq Encoder-Decoder Model	34
Figure 1.11: Transformer Model Architecture	36
Figure 1.12: Seq2Seq Encoder-Decoder with Attention Architecture	37
Figure 1.13: Self-attention Architecture	37
Figure 1.14: Bert Model	39
Figure 1.15: GPT Architecture	40
Figure 1.16: Text-to-Text Transfer Transformer Framework	42
Figure 2.1: Syllable Sequential Labeling Transformer Encoder-Decoder Model	52
Figure 3.1: The Overall Study	54
Figure 3.2: The Sequential Labeling	63

LIST OF TABLES

	<u>Page</u>
Table 1.1: All potential critical characters and their counterparts	1
Table 1.2: Diacritic Letters in Some Latin Languages	4
Table 1.3: Mathematical representation of Markov Chains	13
Table 1.4: Mathematical representation of Hidden Markov Model	14
Table 1.5: LDA Model Steps	20
Table 1.6: Confusion Matrix	46
Table 3.1: The result of Experiment 1	60
Table 4.1: The result of an example set in diacritic correction	67
Table 4.2: Comparing the study of character-based results	68
Table 4.3: Comparing the study for ambiguous words results	70

1. INTRODUCTION

The usage of social media and the internet has increased text production in many languages in the modern period. In addition to English, phonetic letters are found in many other languages that employ the Latin alphabet, including Spanish, German, Croatian, Romanian, Vietnamese, and Turkish. Because some of these phonetic characters—also referred to as diacritical marks—cannot be produced with English keyboards, they might provide difficulties in computer environments. Diacritical letters are frequently omitted from writing by many people. They frequently use the keyboards of their English equivalents. This may result in ambiguity problems when text lacks diacritical characters and "out of vocabulary (OOV)" problems in the field of natural language processing (NLP).

A phrase used in the field of natural language processing is diacritic restoration. Words written partly or fully in American Standard Code for Information Interchange (ASCII) format must be correctly spelled in this process, which is also known as accent restoration, deasciification, or diacritization.

In Turkish language, there are some characters that are not found in English. These characters are called diacritic letters and they include “ç, ı, ğ, ö, ş, ü, Ç, İ, Ğ, Ö, Ş, Ü”. Every letter in Turkish has a unique usage which is independent of its usage in other languages. When it comes to written language, diacritic errors may arise from using some counterpart characters in the English alphabet instead of the diacritic letter characters. Sometimes, errors may occur when English characters are used instead of their Turkish counterparts. In Table 1.1, you can find all the critical characters that could cause a misspelling and their corresponding characters.

Table 1.1: All potential critical characters and their counterparts.

Lower Case Form		Upper Case Form	
Potential Diacritically Incorrect Character	Possible Counterpart In Turkish	Potential Diacritically Incorrect Character	Possible Counterpart In Turkish
c	ç	C	Ç
i	ı	İ	İ

g	ğ	G	Ğ
o	ö	O	Ö
s	ş	S	Ş
u	ü	U	Ü
ç	c	Ç	C
ı	i	I	İ
ğ	g	Ğ	G
ö	o	Ö	O
ş	s	Ş	S
ü	u	Ü	U

Here is an example of a Turkish sentence written in the English alphabet: "Beklenen olumlu tepkiyi alamadı." There are six crucial letters in this statement. This implies that 2^6 distinct sentences with various letter combinations that could be these letters' counterparts can be made. Many of the alternative words in the statement won't have any significance, though. It will be quite simple to determine the correct letters based on the relevant word, as most of the time just one of these possible words will have any significance. The right diacritical letters can be quickly ascertained without the use of an artificial intelligence program by using a morphologic analyzer tool designed for Turkish to identify these significant words. Nonetheless, there are situations in which more than one word that is rebuilt by swapping out one or more crucial letters for their equivalent letters may have significance. We refer to these terms as ambiguous words. The word "olumlu" in the sample sentence above can be changed to either "ölümlü" (meaning "mortal") or "olumlu" (meaning "positive"). However, "olumlu" should be changed to "olumlu" and not "ölümlü" in the sentence "beklenen olumlu tepkiyi alamadı." In particular, an artificial intelligence neural network that can make the correct decision is required to overcome this challenge. Because it is possible to demonstrate the ability to determine the correct word among ambiguous words by determining whether or not the word is meaningful in the sentence, we attempted to measure the semantic success of the trained artificial neural network in this study by evaluating it for ambiguous words.

Studies that just use n-grams are statistical and do not provide any contextual information. This methodology has been used in several related Turkish studies. Decisions made using these statistical models may be semantically incorrect since they function without regard to meaning. Context-aware studies have therefore gained

significance. Neural network models are very effective at extracting contextual information from raw data, according to recent studies. Therefore, context-aware studies usually use the word-based approach rather than the character-based approach. In contrast, letters lack meaning, whereas words do. character-based studies do, however, have certain benefits. Specifically, with character-based systems, the quantity of letters limits the vector space under study. However, when dealing with word-based data in Turkish, it is possible for the vocabulary size—which serves as a vector capacity indicator—to surpass hundreds of thousands of words. All the benefits are combined while working with syllables. Even though there aren't many thousands of syllables in the researched area, syllables' ability to convey meaning is advantageous. Working with syllable-based models makes more sense, particularly when the language structure employed gives the syllables a lot of meaning. Turkish is an agglutinative language as well, with rich meaning elements in its syllables. As a result, this study developed a syllable-based seq-to-seq neural network model.

This study also advises introducing some constraints and rules into the training data to ensure that the trained model is aware of these language structure-related restrictions and rules. Imagine a neural network that can identify a text as a sequence. This text can be arranged alphabetically, syllable by syllable, or word by word into a sequence. If the letter-by-letter sequential structure option is used, the text will be divided into individual letters and supplied to the model as a letter sequence. However, with this structure, the model doesn't know when a word starts or ends, how many letters a word has, how many syllables a phrase contains, etc.

There are various approaches to solving this challenge with a syllable-based data structure. For instance, depending on where a syllable appears in a word, it can be described using a variety of entity vectors. Thus, a particular vector is used to indicate circumstances in which a particular syllable occurs first in the word. The network learns this syllable, which has the word beginning feature, because other usages of this syllable in words will be expressed with different vectors. The relationship between two distinct expressions of the same syllable is broken, though, because these vectors express the same syllable but use separate vectors. There is a statistical flaw in the model that will make learning more challenging. Therefore, using this strategy to extend syllable entities would be unwise. Nonetheless, there is a way around this challenge. Sequential statistics provides the basis for how we know seq-to-seq models

operate. Naturally, we may transfer restrictions and rules to this sequential statistic structure without compromising the statistical integrity of the original entities. By incorporating the sequential entities representing the restrictions and rules into the sequential entities representing the texts, we were able to add these rules to the training set. The neural network model was able to comprehend the semantic features more fully thanks to the training data that was designed.

To sum up, our solution approach is predicated on two fundamental concepts. Firstly, because Turkish is highly sensitive to syllable structure, the seq-to-seq bidirectional transformer model was created as a syllable-based model. Second, a sequential structure including a few rules and restrictions was applied to the training set. By providing support to the statistical framework, these rules and restrictions allowed the network to learn more effectively and accomplish state-of-the-art performance.

1.1. Diacritic Restoration

Diacritics Restoration, sometimes referred to as accent restoration, diacritization, or deasciification, is the act of accurately adding diacritical marks back into text that was originally written without them. The work of adding missing diacritical marks to text written in languages with diacritically marked letters in their orthography—where the diacritical marks are substituted with the matching “Latinized grapheme”—is another common definition of DR [Suppa, 2018].

There are several explanations for why texts lack diacritical marks. For instance, people who use social media solely to share ideas and don't care to write, or those who don't have a designated keyboard. Additionally, a lot of morphological, lexical, and phonological information may be lost while translating texts from one language to another. Hence, DR is a more important task for real-world and NLP systems.

Table 1.2: Diacritic Letters in Some Latin Languages [Web 2].

Language	Diacritics
Afrikaans	á, â, é, è, ê, ë, í, î, ï, ñ, ó, ô, ö, ú, û, ü, ý
Albanian	ë, ç
Dutch	á, à, ä, é, è, ê, ë, í, î, ï, ó, ô, ö, ú, û, ü, ç

English	none
Estonian	ä, ö, ü, õ, š, ž
Finnish	å, ä, ö, š, ž
German	ä, ö, ü, ß
Hungarian	á, é, í, ö, ó, ü, ú, ű
Italian	à, è, é, ì, í, ò, ó, ù, ú
Polish	ą, ć, ę, ł, ń, ó, ś, ź, ż
Turkish	ç, ı, i, ğ, ö, ş, ü
Welsh	â, ê, î, ô, û, ŵ, ŷ, ä, ë, ï, ö, ü, w̃, ỹ, á, é, í, ó, ú, w̄, ý, à, è, ì, ò, ù, ẁ, ÿ

The primary difficulty with this job is the ambiguity that occurs while choosing the best word variant based on context. Eliminating diacritical marks results in new vocabulary groups, which further increases job difficulty. Sentences can have their meaning drastically changed by missing diacritical marks [Nga et al., 2019].

In languages where they are used, diacritical marks are important because they affect how sentences are understood. Sentences written without punctuation may be unclear. Nonetheless, a lot of individuals type without using diacritical marks because it's quicker, more practical, or their gadget doesn't support them. Unfortunately, this can make natural language processing (NLP) activities like sentiment analysis, machine translation, and question-answering systems more challenging to process the text. Thus, to process NLP-related activities further, it is essential to restore diacritics [Náplava et al., 2021].

DR studies have become important in NLP, and there are many different methods available. If we categorize the studies;

- Language-dependent or language-independent studies.
- Character, word, or syllable-based studies on the structure of the language in morphologic approaches.
- Statistical, hybrid, or deep learning methods.

On the other hand, we will evaluate the research regardless of language. The categories mentioned above will also be analyzed. What is important for this study is especially

the developments in Turkish. This study is in Turkish. Therefore, it will be evaluated as Turkish studies and other language studies.

1.1.1. Diacritic Restoration in Turkish

Previous studies included statistical studies and character-based approaches, and this does not include any contextual context. It has been observed that n-grams when used with statistical methods in studies that depend on the Turkish language, yield successful results.

Akgül et al. indicate that machine learning produced better results based on the outcomes of statistical approaches (n-grams). The 1-gram, 2-gram, and 3-gram results were very identical. They explained the reason for this, n-grams only employ neighborhood information [Akgül et al., 2013].

Yıldırım et al. developed a method to generate all possible variations of diacritics and their ASCII equivalents for problematic Turkish words that contain letters that can be difficult to represent. They first verify whether a valid word exists in the candidate set, and if there are multiple intra-word (IV) candidates, they choose the one that occurs most frequently [Yıldırım et al., 2015].

Ozer et al. looked into Turkish tweet data. Using the word2vec model, candidate words were found using the SkipGram model in this study, which had a 94.20% success rate [Ozer Z. et al., 2018].

Trmorph is a potent two-level morphological analyzer for Turkish that was studied by Cöltekin [Cöltekin, 2010]. This work offers the first publicly available Turkish language analyzer and is readily adaptable to different uses. It is implemented with Stuttgart finite state transducer tools (SFTS) that use FSA.

Adali and Eryigit studied social media data. Language validators and discriminative sequential classifiers were employed. With 97.06% accuracy, they achieved a successful outcome [Adali et al., 2014].

Özge et al. claim that not all studies include contextual information in character representations in the Turkish language. They used sequence-to-sequence modeling, which has not been studied in Turkish so far [Özge et al., 2022].

In this study, to believe that conducting a syllable-based study in the Turkish language could lead to more accurate and effective results compared to word-based and

character-based studies. We work on syllable-based context-aware sequence to sequence in the Turkish language sounds like a promising approach to achieving this goal. It's exciting to see the potential advancements that can be made through language research and analysis. A study with a syllabic approach has not been conducted on Turkish yet, and it will be a first in this sense.

1.1.2. Diacritic Restoration in Other Languages

N-gram models, Bayesian classifiers, and a combination of both in the form of decision lists were employed in one of the earliest publications in this field, [Yarowsky, 1999], which already showed about 90% accuracy with a strong vocabulary.

Numerous works focus specifically on deep learning and sequence-to-sequence methods for diacritic restoration. Sequence-to-sequence modeling for the DR problem in many languages has been offered in some of the research like the study of Alkhatlan et al., the study of Masala et al., and the study of Orife, etc.

The Arabic language was researched by Alkhatlan et al. Using LSTM (Long-Short-Term Memory) models, they developed an RNN (Recurrent Neural Network) model. Additionally, they included GRU (Gated-Recurrent Units) in the models. Their Diacritic Error Rate (DER) ranged from 2.32% to 0.19% [Alkhatlan et al., 2020].

The other study is based on Sequence-to-sequence modeling that was studied by Orife. Orife studied in the Yoruba language and focused on Automatic Diacritic Restoration (ADR) and used RNN Encoder and Decoder architecture. 95.4% test accuracy and 98.5% training accuracy results were noted [Orife, 2018].

Whether language-dependent or language-independent models are proposed, more recent work has turned to deep-learning approaches for diacritic restoration.

A similar method was used by Masala et al., who predicted a Romanian BERT model based on pre-given text embeddings. Their model uses a multi-layer bidirectional model called RoBERT. 12.6 GB of data involving RoWiki, Oscar, and RoTex were gathered by them. They produced a baseline research on the language of Romanian [Masala et al., 2020].

Hung et al. conducted a study on deep-learning approaches using LSTM based on a character-based model. The study involved 150 MB of data, comprising 1505000 sentences, with a corrected word accuracy of 95% [Hung et al., 2018].

Naptava et al. conducted a study on a deep-learning approach using BERT, evaluating its results on 12 languages with diacritics including Vietnamese, Romanian, Latvian, Czech, Polish, Slovak, Irish, Hungarian, French, Turkish, Spanish, and Croatian. The model utilizes a fully connected feed-forward neural network with a softmax layer. BERT model processes text without diacritics and tokenized subwords, resembling syllable-based models. The batch size consists of 2048 sentences, and character-based representations are primarily used to accurately recover accent marks [Náplava et al., 2021]. The study compared its results with those of Naptava et al. showing an overall corrected diacritics accuracy of 95% for the 12 languages [Náplava et al., 2018].

Naplava et al. conducted a study on a character level using a bidirectional RNN model with LSTM units. They utilized 30% of Wikipedia data and 20% of Twitter data, totaling almost 30MB of data. The results were assessed with alpha-word accuracy for 12 different languages [Náplava et al., 2018].

It's interesting to see that attention-based studies can be conducted in various ways depending on the language and the unit of analysis chosen. While most studies are either word-based or character-based, Nguyen et al. investigated five letter-based feature sets: learning from letters, semi-syllables, syllables, words, and bi-grams. With a 94.7% success rate, their letters produced superior results [Nguyen et al., 2010].

1.2. Training Models

Upon reviewing the literature, we find that statistical models and structures like n-gram are frequently employed in earlier research. Deep learning models are being employed more frequently nowadays, nevertheless, given the quick growth and learning of artificial intelligence models. These two types of training models will be covered separately in 1.2.1 and 1.2.2.

1.2.1. Statistical Models

Language models, or LMs for short, are word sequences or models that assign probabilities to words in the future. Why would we want to know what LM will say next? It turns out that new huge language models have evolved, and that word prediction alone is the basis for modern NLP training. These are techniques based on statistics.

A word- n -gram language model is a language model that is solely based on statistics. It has been replaced by models based on recurrent neural networks, which have been replaced by models with massive language representations. Its foundation is the idea that the likelihood of a word occurring next in a series solely depends on a predetermined window of words that have come before it. The term "bigram model" refers to a situation in which only one prior word was taken into account; "trigram model" denotes two words; and " n -gram model" denotes $n - 1$ words. To indicate the beginning and conclusion of a phrase, unique tokens called *and* and *were* were introduced [Web 1].

Character-level and word-level diacritization are the two most widely utilized approaches. Word-level techniques are more accurate, but they don't account for non-standard language forms and need costly resources like dictionaries, language models, and monolingual texts. They are therefore less appropriate for languages with fewer resources. However, character-level techniques are more adept at managing words that are not part of the vocabulary, which makes them a better choice when it comes to criticizing texts written in non-normative languages, such as internet comments, forum debates, and social media posts, where the issue of missing diacritics is more common [Stankevičius et al., 2022].

Most word-level statistical methods make use of probabilistic n -gram language models that have already been trained [Náplava et al., 2021]. Large monolingual corpora that provide the likelihood of coming across a specific word sequence in a text are utilized to train these models. These models' resilience is reliant on the quantity and diversity of training data. Better language modeling and reduced confusion are achieved with a higher-order n -gram model. Higher orders, however, can result in sparseness and zero conditional probabilities because they need a large amount of data for training. The closed-world assumption, which underlies these models, holds that any words that are absent from the language model are nonexistent. Smoothing algorithms are therefore essential for handling unknown words and assigning non-zero probabilities.

Due to sparseness arising from uncommon pairings, larger n numbers are generally less favorable for languages with unrestricted word order in a sentence and more difficult to store and compute. Consequently, in order to make language models less sparse and more resilient, pruning is frequently needed [Stankevičius et al., 2022].

1.2.1.1. N-gram Models

A basic approach in NLP, n-grams are a more straightforward and effective technique. The frequency or the number of times a word appears in a sentence is the statistical calculation of a word in a sentence.

The probability of a word given history h is calculated as $P(w|h)$.

Relative frequency counts can be used to determine this probability. Using a big corpus, count the instances in which *its water is so transparent that* and the number of times it follows that. "How many times did the w word come after the times we saw the date h ?" is what this indicates. The following will be the response to the query [Jurafsky et al., 2024].

$$P(the|its\ water\ is\ so\ transparent\ that) = \frac{C(its\ water\ is\ so\ transparent\ that\ the)}{C(its\ water\ is\ so\ transparent\ that)} \quad (1.1)$$

When the chain rule of probability is applied to this probability;

$$P(A, B) = P(A|B), P(B) \quad (1.2)$$

$$P(A, B, C) = P(A|B, C).P(B, C) = P(A|B, C).P(B|C).P(C) \quad (1.3)$$

In general formulation;

$$\begin{aligned} &P(X_1, X_2 \dots X_n) \\ &= P(X_1).P(X_2|X_1).P(X_3|X_1, X_2) \dots P(X_n|X_1 \dots X_{n-1}) \end{aligned} \quad (1.4)$$

$$P(W_1^n) = P(W_1).P(W_2|W_1).P(W_3|W_1^2) \dots P(W_n|W_1^{n-1}) \quad (1.5)$$

$$\prod_{k=1}^n P(W_k|W_1^{k-1}) \quad (1.6)$$

In the unique circumstance of a bigram, or 2-gram;

$$\prod_{k=1}^n P(W_k|W_{k-1}) \quad (1.7)$$

Maximum likelihood estimation, or MLE, is a simple and obvious method of estimating probabilities. Given a preceding word, W_{n-1} , we may calculate the bigram probability of a word W_n counting its bigram $C(W_{n-1}|W_n)$ and normalizing it using the total of all bigrams that have the same initial word, W_{n-1} :

$$P(W_n|W_{n-1}) = \frac{C(W_{n-1}|W_n)}{C(W_{n-1})} \quad (1.8)$$

For n-gram parameter estimation with MLE;

$$P(W_n|W_{n-N+1:n-1}) = \frac{C(W_{n-N+1:n-1}W_n)}{C(W_{n-N+1:n-1})} \quad (1.9)$$

Putting a language model into an application and measuring how much it improves performance is the best way to evaluate the effectiveness of a language model. External evaluation is the term for this type of comprehensive external evaluation. External evaluation is the only method available to determine whether a particular language model (or any component) change will actually help the task at hand. In light of this, n-gram language models used in some applications such as machine translation and speech recognition can be evaluated [Jurafsky et al., 2024].

By using the voice recognizer or machine translator twice, once for each language model, you can compare the performance of two potential language models and determine which one produces transcriptions that are more accurate. Unfortunately, it is frequently highly expensive to run large NLP systems end-to-end. Rather, it would be helpful to have a statistic that could be used to assess possible internal language model enhancements quickly. A metric used for internal evaluation assesses a model's quality without regard to its implementation. We present perplexity, the industry-standard internal metric for evaluating language model performance, in the following section [Jurafsky et al., 2024].

In order to assess neural language models, we also use perplexity, which is one of the most significant metrics in natural language processing. The perplexity (abbreviated as PP or PPL occasionally).

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(W_i|W_{i-1})}} \quad (1.10)$$

As we previously discussed, perplexity depends on the text as well as the language model: given a text W , various language models will have varying levels of perplexity. Perplexity is therefore useful when comparing various n -gram models.

1.2.1.2. Hidden Markov Models (HMMs)

Stochastic modeling approaches like Markov Chains and Hidden-Markov Models are used to represent dynamic systems whose future state depends on their current state. When producing natural language, the Markov chain—which produces a series of words to form a whole sentence—is widely employed. The named-entity recognition and part-of-speech tagging processes use the hidden Markov model, which predicts hidden tags based on words that are observed [Almutiri et al., 2022].

A basic understanding of random, stochastic, and deterministic processes is necessary in order to comprehend the Markov model. Furthermore, it must be defined what state, state-space, and general processes mean. Generally used to represent physical environments, a "state" is made up of a set of variables, each of which is assigned a value [Almutiri et al., 2022]. For example, we have weather states such as sunny rainy, or cloudy. Processes alter states within the state space that encompasses all possible states; they characterize the transition between states in Figure 1.1 [Myers et al., 2017].

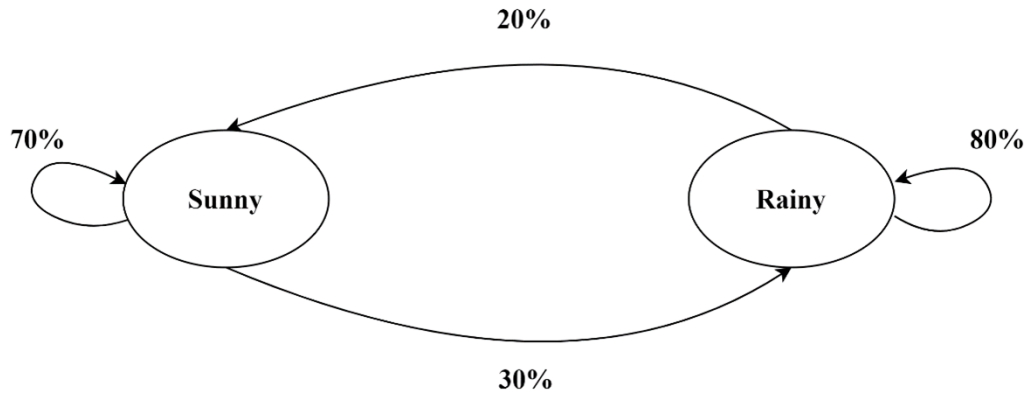


Figure 1.1: An example of weather prediction using Markov Chains [Myers et al., 2017].

The following can be a weather prediction if we choose probability at random: In the event that the day is sunny, there is a 30% chance that it will rain the next day and a 20% chance that it will rain the day after. A sunny day has a 70% probability of continuing to be sunny the following day, whereas an overcast day has an 80% likelihood of continuing to be rainy the day after it ends.

Examining the mathematical computation, defined the states that $q_1, q_2, \dots, q_i$ takes into account a range of state variables. The Markov assumption about the probabilities of this sequence is expressed in a Markov Markov model, which suggests that only the present matters in future predictions, not the past [Jurafsky et al., 2024].

$$P(q_i = a | q_1 \dots q_{i-1}) = P(q_i = a | q_{i-1}) \quad (1.11)$$

The following elements together define a Markov chain formally:

Table 1.3: Mathematical representation of Markov Chains.

$Q = q_1, q_2 \dots q_n$	A group of N states
--------------------------	---------------------

$A = a_{11}, a_{12} \dots a_{N1} \dots a_{NN}$	A transition probability matrix, where each a_{ij} denotes the likelihood of transitioning from state i to state j, $\sum_{j=1}^n a_{ij} = 1 \forall_i$
$P = p_1, p_2 \dots p_n$	An initial probability distribution among states. The likelihood that the Markov chain will begin in state i is denoted by p_i. There are states j that cannot be beginning states because $p_j = 0$. $\sum_{i=1}^n p_i = 1$

When determining the probability of a series of observable occurrences, a Markov chain might be helpful. Nevertheless, the events that fascinate us are frequently hidden from our direct view. For instance, part-of-speech markers are not typically observed in texts. Instead, words are presented to us, and we must deduce the tags from the word order. The tags are not observed, which is why we refer to them as hidden.

Using a Hidden Markov Model (HMM), we may discuss both hidden events—such as part-of-speech tags—that we consider to be causative elements in our probabilistic model and observable events—such as words we see in the input. [Baum et al., 1966] developed HMM first.

Table 1.4: Mathematical representation of Hidden Markov Model [Jurafsky et al., 2024].

$Q = q_1, q_2 \dots q_n$	A group of N states
$A = a_{11}, a_{12} \dots a_{N1} \dots a_{NN}$	A transition probability matrix, where each a_{ij} denotes the likelihood of transitioning from state i to state j, $\sum_{j=1}^n a_{ij} = 1 \forall_i$

$B = b_i(o_t)$	A series of emission probabilities , or observation likelihoods , that individually represent the likelihood of an observation
$O = o_1, o_2 \dots o_n$	A series of O observations , all drawn from a certain vocabulary. $V = v_1, v_2 \dots v_n$
$P = p_1, p_2 \dots p_n$	An initial probability distribution among states. The likelihood that the Markov chain will begin in state i is denoted by p_i . There are states j that cannot be beginning states because $p_j = 0$. $\sum_{i=1}^n p_i = 1$

The state that generated the observation q_i alone determines the probability of an output observation o_i ; no other states or observations are involved.

When applying observations on Markov's assumption;

$$P(o_i | q_1 \dots q_i \dots q_T, o_1 \dots o_i \dots o_T) = P(o_i | q_i) \quad (1.12)$$

1.2.1.3. Naive Bayes Classifier

The Naive Bayes classifier is widely employed in a variety of classification tasks and is not limited to the field of Natural Language Processing, despite its name and the naivete required in its creation [Suppa, 2018].

[Mosteller et al., 1964] were the first to use the concept of Bayesian inference, which dates back to the work of [Bayes, 1763]. The idea behind Bayesian classification is to convert Eq. 1.13 into different probabilities with some beneficial characteristics by applying Bayes' rule.

Naive Bayes is a probabilistic classifier, which means that given a document d, it will yield the class $\hat{c}$ with the largest posterior probability among all classes $c \in C$ [Jurafsky et al., 2024].

$$\hat{c} = \operatorname{argmax}_{c \in C} P(c|d) \quad (1.13)$$

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \quad (1.14)$$

When substituting Eq. 1.13 and 1.14;

$$\hat{c} = \operatorname{argmax}_{c \in C} P(c|d) = \operatorname{argmax}_{c \in C} \frac{P(d|c)P(c)}{P(d)} \quad (1.15)$$

The conditional independence assumption that the probabilities $P(f_i|c)$ are independent given the class c can be "naively" multiplied as follows. This is usually referred to as the "Naive Bayes assumption".

$$P(f_1, f_2 \dots f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot \dots \cdot P(f_n|c) \quad (1.16)$$

The following is the final equation for the class that a naive Bayes classifier selected:

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \prod_{f \in F} P(f|c) \quad (1.17)$$

When we apply the word positions in text;

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \prod_{i \in positions} P(w_i|c) \quad (1.18)$$

Generally, causes an underflow of the floating point as n grows. Adding logarithms of probabilities, as opposed to multiplying them as in step 3, is therefore a superior way to do the computation [Cocks, 2011]. When applying log space to boost speed and prevent underflow;

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \sum_{i \in positions} \log P(w_i|c) \quad (1.19)$$

1.2.1.4. Maximum Entropy Models (MaxEnt)

A different probabilistic machine learning framework is called Maximum Entropy modeling, or simply MaxEnt. Since its introduction, the maximum entropy (ME) method has become increasingly well-liked in the field of natural language processing. Moreover, due to its many advantages over alternative statistical techniques, the MLE [Kazama et al., 2005] authors employed it for data sparseness. They suggest using inequality constraints and highlight how the usual ME method is ineffective for enforcing equality constraints. With this approach, they managed to solve data sparseness with MLE.

Multinomial logistic regression is a more commonly used name for MaxEnt to describe how to classify sequences using MaxEnt. The process of giving a label to each element in an array, for instance, giving a series a part-of-speech tag, is known as array categorization or sequence labeling. The Maximum Entropy Markov Model, or MEMM, is the most widely used MaxEnt sequence classifier. It is important to mention non-sequential classification as well. The classification problem involves taking a single observation, identifying its valuable qualities, and using these features to categorize the observation into one of a number of distinct classes. Beyond just classifying an observation and providing a label, a probabilistic classifier also provides the likelihood that the observation belongs to that class. In fact, it provides a probability distribution across all classes for a given observation [Jurafsky et al., 2024].

The sentence's boundaries must be known in order to do sentiment analysis or text classification. A feature is generated using text x as an input. The objective is to choose the class based on the specified feature and weight. Mostly uses MaxEnt as an exponential or log-linear classifier. The calculation is that weights w_i are added to and multiplied by the feature input f_i . The formulation is as follows:

$$P(c|y) = \frac{1}{Z} \exp \left(\sum_i w_i f_i \right) \quad (1.20)$$

1.2.1.5. Conditional Random Fields (CRFs)

A probabilistic framework for labeling and partitioning sequential data based on the conditional approach is provided by conditional random fields [Lafferty et al., 2001] (CRFs). Conditional probability distributions (CRFs) are particularly useful for modeling conditional probability distributions across dependent and independent variables in sequential data labeling tasks. A graphical model for conditional probabilities is called CRF. Given a set of observations, it is used to estimate a sequential labeling structure of hidden states. Because CRFs are conditional, they can relax the independence assumptions that HMMs require in order to provide tractable inference. This gives them an advantage over hidden Markov models. Said another way, CRF offers greater flexibility than HMMs and hidden Markov models.

Furthermore, CRFs do not suffer from the label bias issue [Lafferty et al., 2001], which is a flaw in maximum entropy Markov models (MEMMs) and other directed graph model-based conditional Markov models. CRFs perform better on certain real-world sequence labeling tasks than both MEMMs and HMMs [Lafferty et al., 2001]. The link between a given collection of observations (such as words) and the labels of those data (such as grammatical labels) is modeled using CRF. This model, given a sequence of observations, predicts the most likely order of labels.

According to Lafferty et al. the likelihood of a specific label sequence (y) given an observation sequence (x) is a normalized product of potential functions, with each of the following formulations [Lafferty et al., 2001]:

$$\exp \left(\sum_j \lambda_j t_j(y_{i-1}, y_i, x, i) + \sum_k \mu_k s_k \right) \quad (1.21)$$

$t_i(y_{i-1}, y_i, x, i)$: a feature function that transitions between the labels at locations i and $i-1$ in the label sequence and the complete observation sequence.

$s_k(y_i, x, i)$: an observation sequence and the label at location i together form a state feature function.

$$P(y|x, \lambda) = \frac{1}{Z(x)} \exp \left(\sum_j \lambda_j F_j(y, x) \right) \quad (1.22)$$

$Z(x)$: a normalization factor.

CRFs can be computationally expensive, particularly when complex features and big data sets are used for training. Compared to basic modeling, the training and implementation processes for CRFs are more complicated and demand more experience.

1.2.1.6. Latent Dirichlet Allocation (LDA)

In the domains of text mining and natural language processing, Latent Dirichlet Allocation (LDA) is a powerful topic modeling tool. It is employed to identify invisible subjects inside extensive text collections and categorize texts by these subjects. The idea underlying LDA is that documents display a variety of topics [Blei, 2012]. Because LDA is effective at classification, investigations that have been done on it have produced positive outcomes. For instance, in Negara E. S. et al., (2019), the LDA method of allocation was used to study Twitter text and generate Indonesian tweet topic data modeling. Word indexes, word clusters, and similarity are produced during this research procedure for each of the four distinct Twitter data subjects. When they examined the LDA and LSI model results, they found that the LDA approach produced tweets with 98% more accuracy than the LSI method. Moreover Zou et al., (2014) developed an effective LDA analyzer tool based on the LDA model, and it includes fresh corpus training as well as the analyzer's results.

The Dirichlet distribution is the distribution that is used to derive the distribution of topics within a text. The outputs of the Dirichlet distribution are then utilized in the generative process of LDA to assign words within the document to various themes. The algorithm known as Latent Dirichlet Allocation is based on the observation that topics, the distribution of themes within a document, and the categorization of every word on subjects inside a document are hidden structures [Negara et al., 2019]. Blei D. M., (2012) generates the text generation model in the following two processes;

1. Select a distribution of themes at random.
2. a. Select a topic at random from the distribution of topics in step 1 for every word in the document.
- b. Select a word at random from the matching vocabulary distribution.

The study of Negara et al. has highlighted the steps of the model to help with comprehension of the LDA model [Negara et al., 2019]. If this is categorized in the Table 1.5;

Table 1.5: LDA Model Steps.

Inputs	M: the number of document t: the number of topics β : vocabulary matrix
Outputs	P: Probability distribution of topics for every word in the document.
Method of Steps	1. Select the distribution of topics α. 2. Assign each word W to a topic from t in a document d. 3. For every word W in document d, compute $P(\text{topic } t \mid \text{Document } d)$ for every subject and $P(\text{word } W \mid \text{Topic } t)$ 4. The distribution of β Vocabulary words determines the word W that is chosen for a topic t.

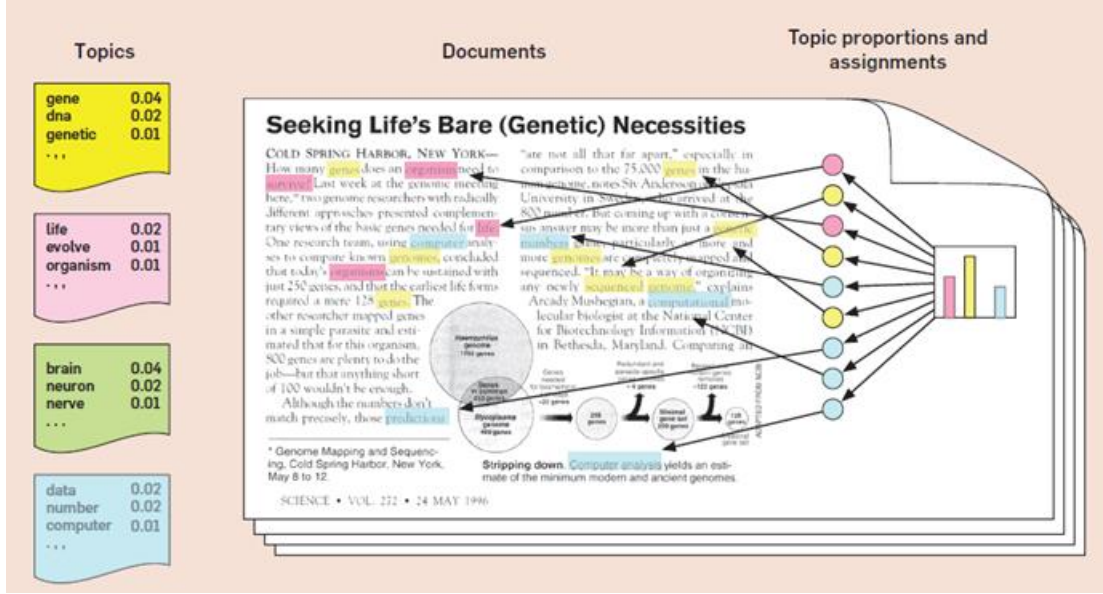


Figure 1.2: LDA Probabilistic Model [Blei, 2012].

Examining the mathematical expression denoted by the study of Blei [Blei, 2012];

Each β_k represents a distribution over the vocabulary, and the topics are $\beta_{1:k}$.

The topic proportions of topic k in document d are denoted by Q_d , $Q_{d,k}$ which represent the topic proportions for the d th document.

For the d th document, the topic assignments are z_d , where $z_{d,n}$ represents the topic assignment for the n th word in the document d .

The words that are noticed in the document d are w_d , where $w_{d,n}$ is the document d word that is n th in length and is a part of the fixed vocabulary.

$$P(\beta_{1:D}, Q_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^K P(\beta_i) \prod_{d=1}^D P(Q_d) \quad (1.23)$$

$$\left(\prod_{n=1}^N P(z_{d,n} | Q_d) P(w_{d,n} | \beta_{1:K}, z_{d:n}) \right)$$

For LDA, posterior computation is defined by the study of Blei [Blei, 2012];

$$P(\beta_{1:D}, Q_{1:D}, z_{1:D} | w_{1:D}) = \frac{P(\beta_{1:D}, Q_{1:D}, z_{1:D}, w_{1:D})}{P(w_{1:D})} \quad (1.24)$$

1.2.2. Deep Learning Models

The subset of machine learning techniques based on neural networks and representation learning is called deep learning.

The topic of diacritics restoration is being solved using novel approaches thanks to the development of Deep Neural Networks (DNNs). Certain methods depend on word embeddings, which are contextually capturing learned word representations.

Recent years have seen remarkable progress with neural networks. We can now recognize patterns, sequences, and correlations thanks to the enormous rise in hidden layers and nodes brought about by deep learning. Deep learning has a number of benefits, including: - Its structure is similar to that of the human brain, in contrast to earlier models that used transition probability to find the optimal answer. Due to their state-based design, traditional models may have overlooked structural combinations that weren't included in the reference lexicon. Deep learning, on the other hand, is a vector-based strategy that can successfully overcome this restriction [Hung, 2018].

For instance, State-of-the-art techniques in diacritics restoration, like in all NLP fields, utilize transformer-based models. Multilingual BERT has been successfully used for 12 languages, including Vietnamese, Romanian, Latvian, Czech, Polish, Slovak, Irish, Hungarian, French, Turkish, Spanish, and Croatian. BERT embeddings that are created on the undiacritized text are input into a fully connected Feed-Forward Neural Network (FFNN). The network's output is a set of instructions, or labels, that define the diacritization operation needed for each character of the input token [Stankevičius et al., 2022].

All in all, deep learning techniques are far more effective when used for DR since they are language-independent and consider contextual information.

1.2.2.1. Recurrent Neural Networks (RNNs) and Variants

RNNs are a kind of neural network that are used to identify patterns in data sequences, including time series or natural language. RNNs can retain a hidden state that can

gather data from earlier time steps because they differ from standard feedforward neural networks in that their connections form directed cycles.

The input layer, hidden layer, and output layer are the three layers that typically make up the architecture of an RNN language model. Figure 1.3 depicts its structure model, a cyclic neural network language model that has been expanded over time. The RNN language model allows for the iterative computation of the likelihood of each word appearing under the provided information semantic condition through temporal expansion. The specifics of each layer's calculation process, including the conditional probability of a single word, are outlined in Figure 1.3.

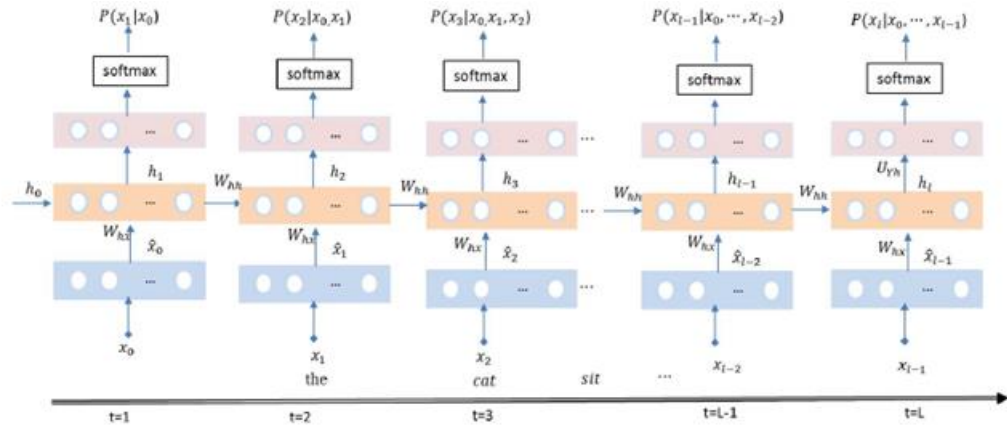


Figure 1.3: Recurrent Neural Networks Language Model [Xiao et al., 2020].

The input layer (x), hidden layer (s), or state or context layer, and output layer (y), make up the recurrent network. Concatenating vector w , which represents the current word, with the output from neurons in context layer s , at time $t - 1$ results in the input vector $x(t)$. Usually, infrequent words are combined into a single token to increase performance. Mathematical formulation is defined by the study of Mikolov et al. [Mikolov et al., 2010];

$$x(t) = w(t) + s(t - 1) \quad (1.25)$$

$$s_j(t) = f\left(\sum_i x_i(t)u_{ji}\right) \quad (1.26)$$

$$y_k(t) = g\left(\sum_j s_j(t)v_{kj}\right) \quad (1.27)$$

The function of Sigmoid activation is $f(z)$;

$$f(z) = \frac{1}{1 + e^{-z}} \quad (1.28)$$

The function of Softmax is $g(z)$;

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \quad (1.29)$$

Examining the advantages of RNN-Based diacritic restoration;

Context Awareness: To properly restore diacritics based on surrounding characters, these models must be able to preserve dependencies and context throughout lengthy sequences.

Flexibility: They can analyze complete phrases or paragraphs since they can handle sequences of different durations of input.

Accuracy: In complicated language applications, advanced variations like LSTM and GRU produce more accurate predictions by mitigating problems like disappearing gradients.

1.2.2.1.1. Long Short-Term Memory Networks (LSTM)

A kind of recurrent neural network (RNN) architecture called Long Short-Term Memory (LSTM) was created especially to manage long-term dependencies in sequential input. Due to problems like vanishing and inflating gradients, traditional RNNs have trouble understanding long-term dependencies, which can make training challenging. By using a more intricate architecture with gates—mechanisms that

control the flow of information—LSTMs are able to address these problems.

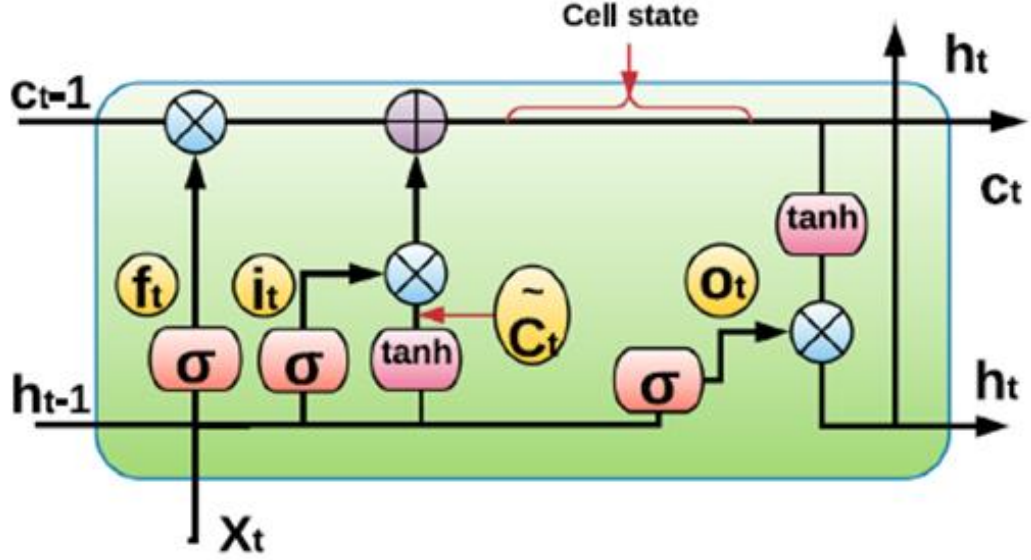


Figure 1.4: LSTM Cell [Hamad et al., 2021].

Cell State (C_t): This is the network's memory, which stores data over various time steps.

Hidden State (h_t): This shows the LSTM's output at a specific time step.

Input Gate: determines the amount that should be added to the cell state as new information from the previous hidden state and the current input.

Forget Gate: determines which data should be deleted from the cell state.

Output Gate: decides what data from the cell state needs to be sent to the following time step.

The LSTM model and formulation provided in the study by Yao et al. are defined by the following equations [Yao et al., 2018]:

$$\text{Forget gate: } f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1.30)$$

$$\text{Input gate: } i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \quad (1.31)$$

$$\text{Cell candidate: } \tilde{C}_t = \tan(W_C \cdot [h_{t-1}, x_t] + b_c) \quad (1.32)$$

$$\text{New cell state: } C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \quad (1.33)$$

$$\text{Output gate: } o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (1.34)$$

$$\begin{array}{ll} \text{New hidden} & h_t = o_t \times \tan C_t \\ \text{state:} & \end{array} \quad (1.35)$$

The sigmoid function, the input, forget, and output gates are represented by the values σ , i_t , f_t , and o_t , respectively, and the candidate and new cell values at time t are represented by the values $\tilde{C}_t$ and C_t . Recall that the dimensions of each of these vectors match the value of the hidden layer, h .

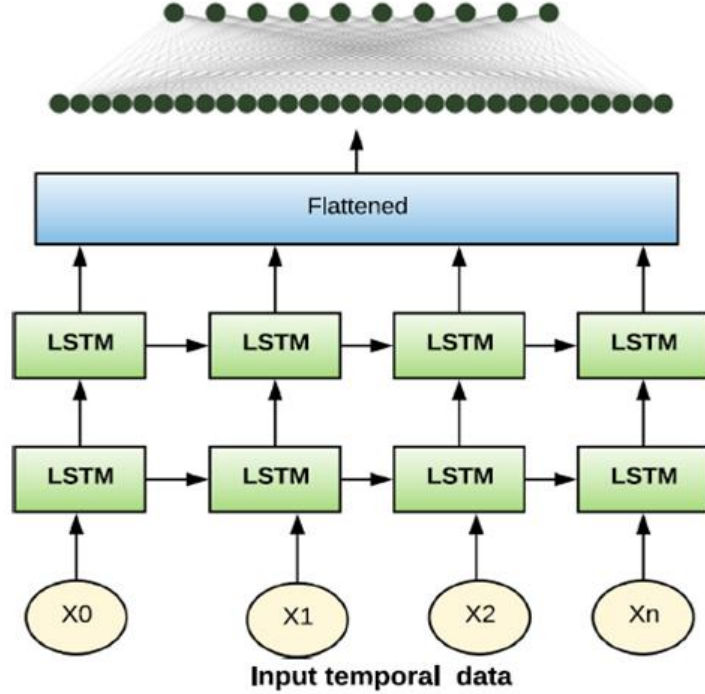


Figure 1.5: LSTM Architecture [Hamad et al., 2021].

Benefits of Diacritic Restoration with LSTM;

Context Awareness: In order to correctly restore diacritical marks, LSTMs must be able to comprehend the context of words and letters.

Long-Term Dependencies: They have the capacity to learn dependencies across extended sequences, which enhances the accuracy of restoration for languages with intricate diacritic rules.

Flexibility: LSTMs can analyze full phrases or paragraphs since they can handle sequences of different lengths.

1.2.2.1.2. Bidirectional Long-Short Term Memory (BiLSTM)

A sequential processing technique called BiLSTM consists of two Long-Short-Term memory, the first of which can receive information in one direction (forward direction) and the second in the opposite direction (backward direction). The network's information availability is greatly improved by bidirectional long-short-term memory, which optimizes the context for the algorithms.

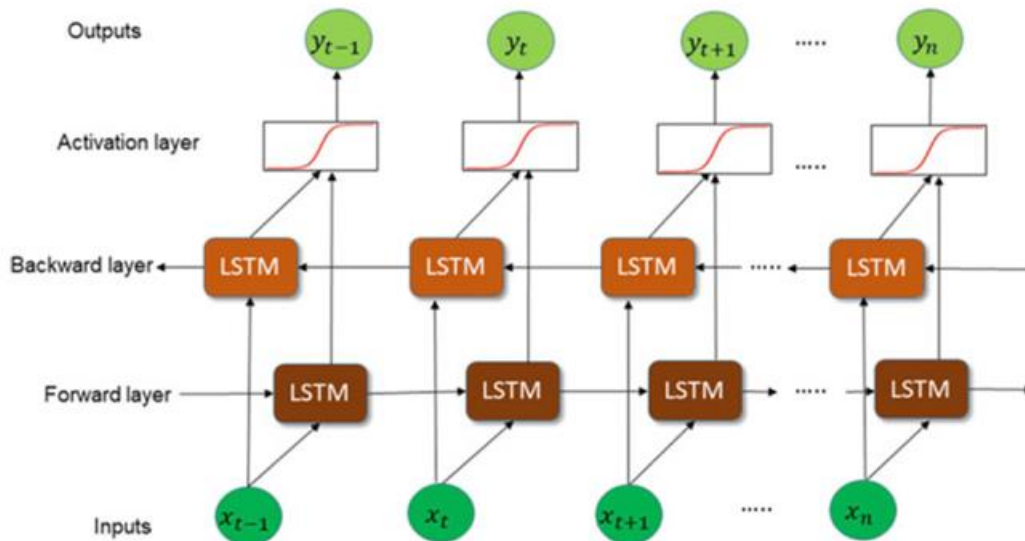


Figure 1.6: BiLSTM Architecture [Al-Qerem et al., 2024].

The BiLSTM in the suggested method is made up of two independent LSTMs that can combine the information originating from the two ways after summing up the information from both directions of a sentence. In particular, at every time t , the input word embedding x_t and the prior hidden vector fh_{t-1} are used by the forward LSTM to calculate the hidden vector fh_t , while the input word embedding x_t is used by the backward LSTM to compute the hidden vector bh_{t-1} . Subsequently, the BiLSTM model's final hidden vector is created by merging the forward hidden vector (fh_t) and the backward hidden vector (bh_t). The BiLSTM model exhibits dependency between the parameters of two opposing directions, despite sharing a sentence's word embedding. The fundamental architecture of the BiLSTM model is shown in Figure 1.6, where n represents sentence length and $\{w_1, w_2, \dots, w_n\}$ indicates the word vector. The forward hidden vector is represented by $\{fh_1, fh_2, \dots, fh_n\}$, and the backward hidden vector is represented by $\{bh_1, bh_2, \dots, bh_n\}$. The vector that is joined by fh_n and bh_n is denoted by h_n .

Equation displays the final hidden vector h_t of the BiLSTM.

$$h_t = [fh_f, bh_f] \quad (1.36)$$

Benefits of Using BiLSTM for Diacritic Restoration;

Bidirectional Context: BiLSTMs grasp more thorough context by processing sequences in both directions, which is essential for precisely positioning diacritics.

Increased Accuracy: BiLSTMs usually perform better than unidirectional LSTMs in jobs where comprehension of the complete context surrounding each element in a sequence is necessary.

Flexibility: BiLSTMs are appropriate for processing sentences or paragraphs since they can handle sequences of different lengths.

1.2.2.1.3. Gated Recurrent Units (GRUs)

The GRU lacks an output gate, making it similar to an LSTM with a forget gate but with fewer parameters. GRU cell is shown in Figure 1.7.

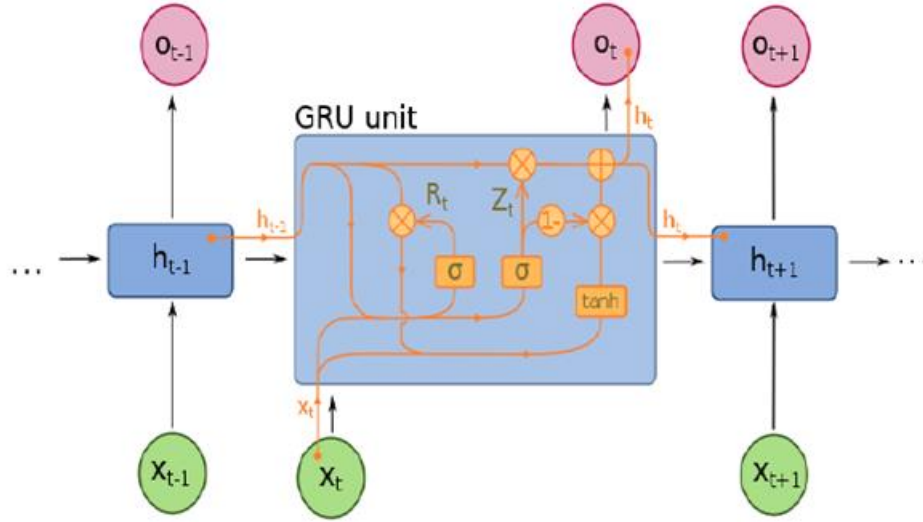


Figure 1.7: GRU Cell [Santur, 2019].

$$\begin{array}{ll} \text{Update gate:} & z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \end{array} \quad (1.37)$$

$$\begin{array}{ll} \text{Reset gate:} & r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \end{array} \quad (1.38)$$

$$\begin{array}{ll} \text{Candidate activation:} & \tilde{h}_t = \tanh(W \cdot [r_t \otimes h_{t-1}, x_t]) \end{array} \quad (1.39)$$

$$\begin{array}{ll} \text{Output gate:} & h_t = (z_t \otimes h_{t-1}) + (1 - z_t) \otimes \tilde{h}_t \end{array} \quad (1.40)$$

In general, the formulations show where W_z , W_r , and W are trainable weight matrices, and x_t , h_t , z_t , and r_t stand for the input, hidden-state, update gate, and reset gate vectors, respectively. Its output is also its hidden state. σ is sigmoid function, $\tanh$ is activation function.

Benefits of Diacritic Restoration with GRUs;

More Basic Architecture: Compared to LSTMs, GRUs contain fewer parameters, which can facilitate their implementation and speed up training.

Effective Context Capture: GRUs are good at capturing sequences with long-term dependencies, even though they are simple to use.

Flexibility: GRUs are appropriate for processing sentences or paragraphs since they can handle sequences of different lengths.

1.2.2.2. Convolutional Neural Networks (CNNs)

In deep learning, CNN is the most well-known model. CNNs are widely used in many domains, including computer vision. CNNs are structured similarly to traditional neural networks, with inspiration drawn from neurons found in the brains of humans and animals. More precisely, the visual cortex in a cat's brain is made up of a complicated series of cells, which the CNN simulates [Alzubaidi et al., 2021].

The CNN architecture, as seen in Figure 1.8, is composed of several layers.

Convolutional Layer: The convolutional layer is the most important part of the CNN design. It is comprised of an assortment of convolutional filters, also referred to as kernels. These filters are used to convolve the input image, which is expressed as N-dimensional metrics, to produce the feature map that is output [Alzubaidi et al., 2021].

Pooling Layer: Subsampling the feature maps is the primary function of the pooling layer. The convolutional operations are followed to create these maps. Put otherwise, this method reduces the size of large-scale feature maps in order to produce smaller feature maps. Simultaneously, it preserves most of the dominating data (or characteristics) at each stage of the pooling process. The stride and the kernel are first size-assigned prior to the pooling procedure, just like in the convolutional operation. Numerous kinds of pooling techniques can be applied in different pooling layers. Tree pooling, gated pooling, average, min, max, global average pooling (GAP), and global max pooling are some of these techniques [Alzubaidi et al., 2021].

Activation Function: The fundamental job of all kinds of activation functions in all kinds of neural networks is to map the input to the output. The computation of the weighted summation of the neuron input and its bias (if any) yields the input value. This indicates that by producing the appropriate output, the activation function decides

whether or not to fire a neuron in response to a specific input. The most popular activation functions are Sigmoid, Tanh, and ReLU [Alzubaidi et al., 2021].

Fully Connected Layer: typically found at the very top of a CNN design. The so-called Fully coupled (FC) technique is used in this layer, where every neuron is coupled to every other neuron in the layer above. It serves as the classifier for CNN. Being a feed-forward ANN, it operates on the same principles as a traditional multiple-layer perceptron neural network. The final pooling or convolutional layer provides the FC layer's input. Following flattening, the feature maps are converted into a vector, which is this input [Alzubaidi et al., 2021].

Loss Functions: Different CNN architecture layer types were covered in the preceding section. Furthermore, the output layer—which is the last layer in the CNN architecture—achieves the final categorization. The predicted error produced by the CNN model over the training samples is computed at the output layer using a few loss functions. The discrepancy between the expected and actual output is shown by this error. The most common using loss functions are Cross-Entropy or Softmax Loss Function, Euclidean Loss Function [Alzubaidi et al., 2021].

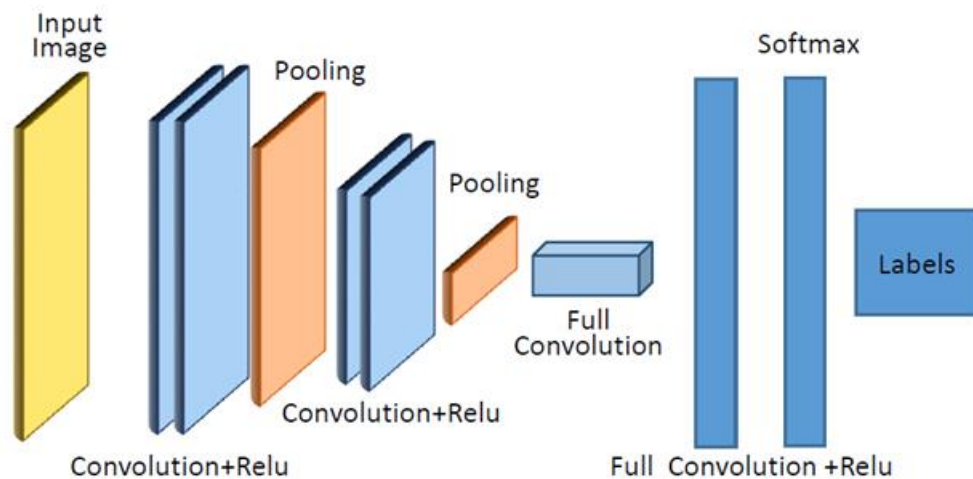


Figure 1.8: CNN Architecture [Afridi et al., 2021].

Similar to the multi-layer perceptron (MLP), a popular kind of CNN has multiple convolution layers ahead of sub-sampling (pooling) layers, with FC layers at the end. Figure 1.9 shows an example of CNN architecture for image categorization.

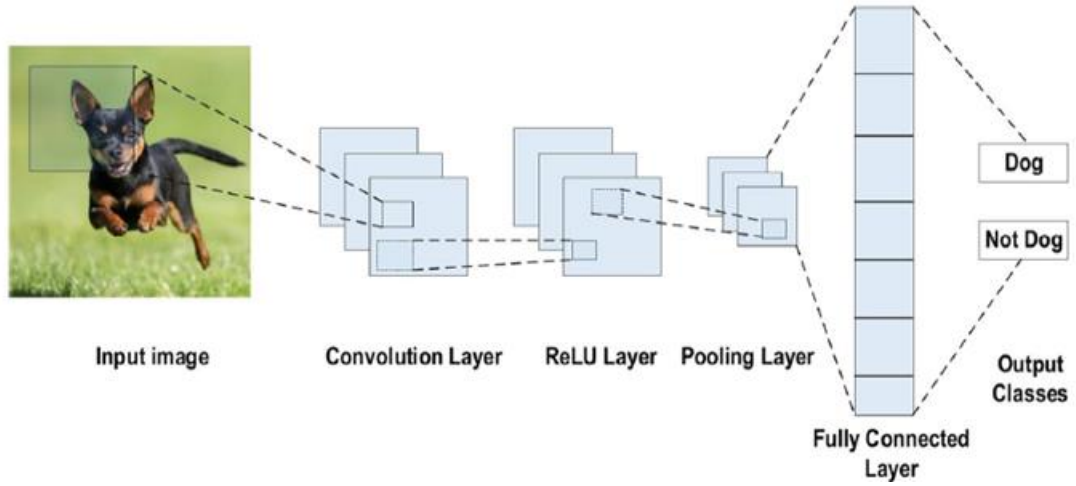


Figure 1.9: A CNN architecture sample used for image classification [Alzubaidi et al., 2021].

The activation function to the output of the convolution layer is calculated as follows;

$$h^k = f(W^k * x + b^k) \quad (1.41)$$

Equation 1.41 defined parameters where each layer's input x , bias b^k , and weight W^k are used to generate k feature maps h^k .

In the context of computer vision, employing CNNs has the following advantages over other conventional neural networks:

- CNN's weight sharing feature is the primary reason to take it into consideration. It lowers the number of trainable network parameters, which improves generalization and prevents overfitting.
- The model output becomes extremely structured and dependent on the retrieved features when the feature extraction and classification layers are learned simultaneously.

Benefits of CNN application for Diacritic Restoration;

Local Pattern Detection: CNNs are essential for applications such as diacritic restoration because they can efficiently extract local patterns and dependencies from text sequences.

Parallel Processing: CNNs are able to train on huge datasets more quickly than certain other models because convolutional operations can be efficiently parallelized.

Dimensionality Reduction: By lowering the dimensionality of feature maps, pooling layers contribute to the computational efficiency of the model.

1.2.2.3. Sequence-to-Sequence (Seq2Seq) Models

A neural network architecture called Sequence-to-Sequence (Seq2Seq) is intended to convert one sequence into another. It is extensively utilized in many different natural language processing (NLP) applications, including diacritic restoration, text summarization, and machine translation. An encoder and a decoder are the two primary parts of the Seq2Seq model.

Seq2Seq's concept consists of two Recurrent Neural Networks (RNNs), called encoder and decoder, respectively, that attempt to predict the subsequent state sequence based on the previous one. Among the longest RNNs were two called LSTM.

Encoder: The encoder is a device that interprets an input sequence and produces a fixed-length context vector, or collection of context vectors.

Usually, it is composed of multiple layers of transformers, Long Short-Term Memory (LSTM) networks, or recurrent neural networks (RNNs).

The encoder processes each piece in the input sequence, capturing the dependencies and context between them.

Decoder: The decoder creates the output sequence, element by element, using the context vector(s) that the encoder produced.

Typically, it is also made up of transformers, LSTMs, or RNNs.

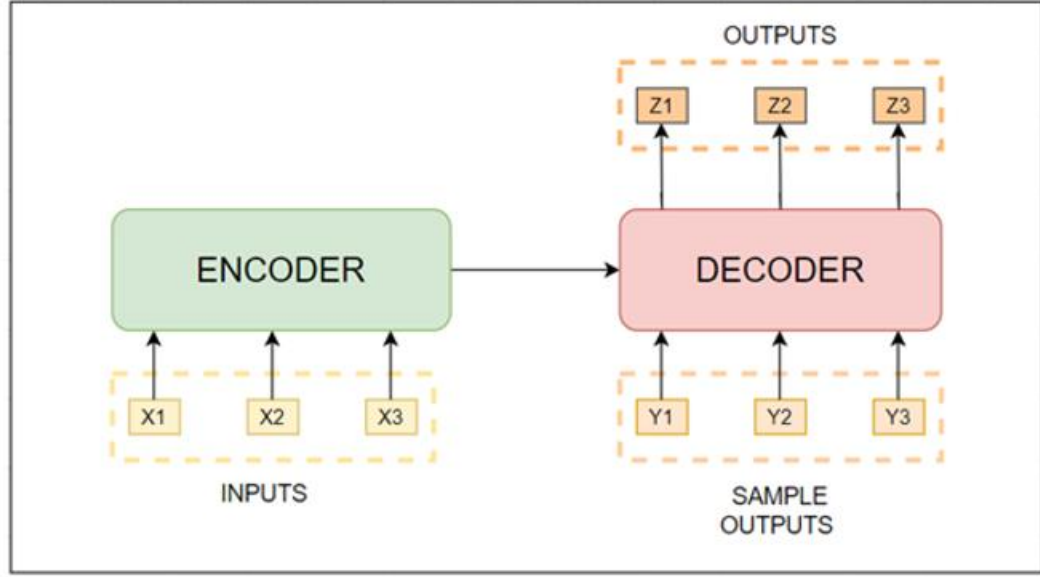


Figure 1.10: Seq2Seq Encoder-Decoder Model.

The decoder predicts the subsequent element in the sequence at each step using the context vector and the output that was previously created.

Word sequences make up the input and output, which are, respectively, $\mathcal{X} = x_1, \dots, x_T$ and $\mathcal{Y} = y_1, \dots, y_T$. Typically, the terms $\mathcal{X}$ and $\mathcal{Y}$ denote the source and target sentences. Sequence-to-sequence (seq2seq) refers to encoder-decoder architectures where both the input and the output are sequences. The creator of the Seq2Seq model is Sutskever et al. [Sutskever et al., 2014]. Maximum likelihood estimation (MLE) can be used to concurrently optimize their parameters over a parallel corpus. This type of instruction is known as end-to-end [Tixier et al., 2018]. The conditional probability $P(y_1, \dots, y_T | x_1, \dots, x_T)$, the representation v is set as the initial hidden state.

$$h_t = \text{sigm}(W^{hx}x_t + W^{hh}h_{t-1}) \quad (1.42)$$

$$y_t = W^{yh}h_t \quad (1.43)$$

$$P(y_1, \dots y_T | x_1, \dots x_T) = \prod_{t=1}^{T'} P(y_t | v, y_1, \dots y_{t-1}) \quad (1.44)$$

Advantages of Sequence-to-Sequence models for Diacritic Restoration;

Contextual Understanding: Sequence-to-Sequence models have the ability to accurately reconstruct the context and semantics of the input sequence.

Sequence Generation: They work effectively on problems where the length of the output sequence may differ from the length of the input sequence.

Flexibility: Seq2Seq models are excellent for languages with varying word lengths and diacritic placements because they integrate attention mechanisms that allow them to dynamically shift their emphasis.

1.2.2.4. Transformer Models

In terms of performance for tasks in both natural language understanding and natural language generation, the Transformer [Vaswani et al., 2017] has quickly surpassed other neural models like convolutional and recurrent neural networks to become the most popular architecture for natural language processing. The design allows for efficient parallel training, captures long-range sequence characteristics, and scales with both training data and model size.

Many transformer-based models have been created and are extensively employed in a variety of natural language processing (NLP) and related fields. Every one of these models has a unique architecture, with variants designed for certain use cases and datasets. Following is a detailed discussion of some of the fundamentally significant transformers.

1.2.2.4.1. Attention is All You Need (Original Transformer)

In the 2017 paper "Attention Is All You Need," Google researchers unveiled a brand-new model they named Transformer [Vaswani et al., 2017]. In a nutshell, this architecture is made up of fully connected layers that are point-wise and self-attentional (refer to Figure 1.11). Transformers consist of a final linear layer, an encoder, and a decoder, just as what we explain for Seq2seq [Montesinos, 2020].

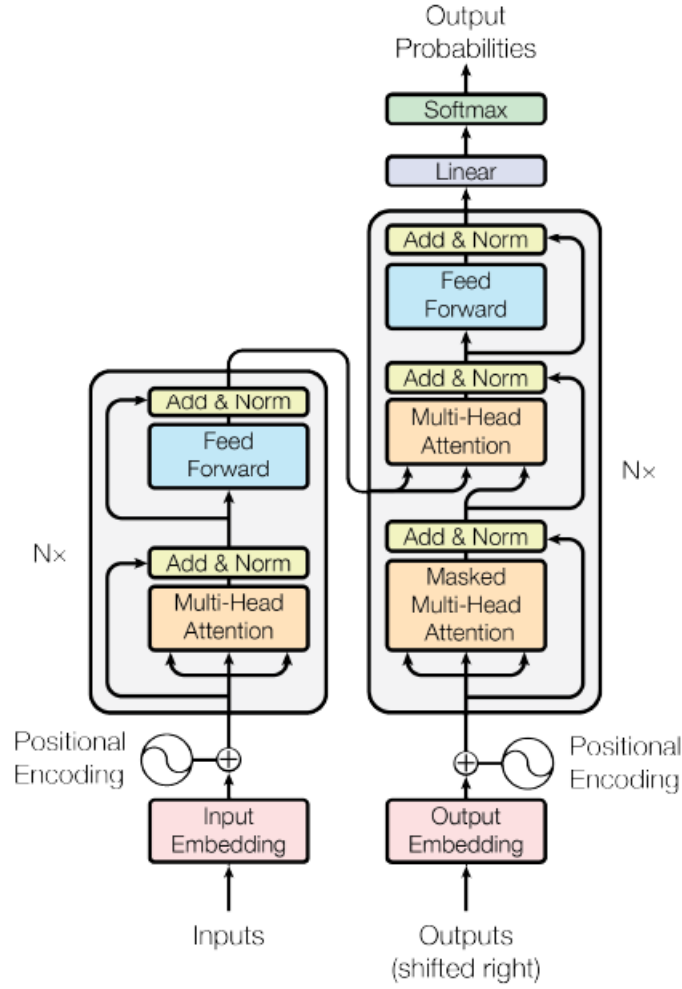


Figure 1.11: Transformer Model Architecture [Vaswani et al., 2017].

The Transformer creates a latent representation $Z = (z_1, \dots, z_N)$ from a sequence $X = (x_1, \dots, x_N)$ using an encoder-decoder structure as its foundation. One element at a time, the output sequence $Y_M = (y_1, \dots, y_M)$ is generated because of this model's autoregressive feature. The Multi-Head Attention layer is utilized by both the Decoder and the Encoder. A single attention layer associates a weighted sum of the values V with a query Q and key K . There is a scaling factor $\frac{1}{\sqrt{d_k}}$ due to technical reasons.

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1.45)$$

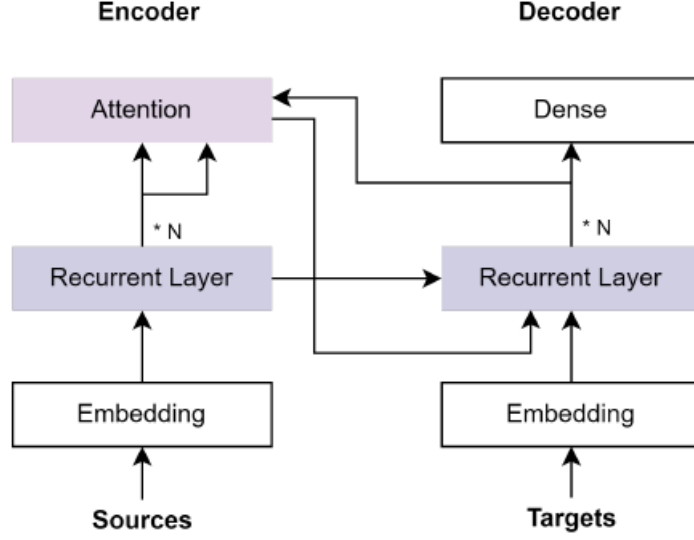


Figure 1.12: Seq2Seq Encoder-Decoder with Attention Architecture [Rahali et al., 2023].

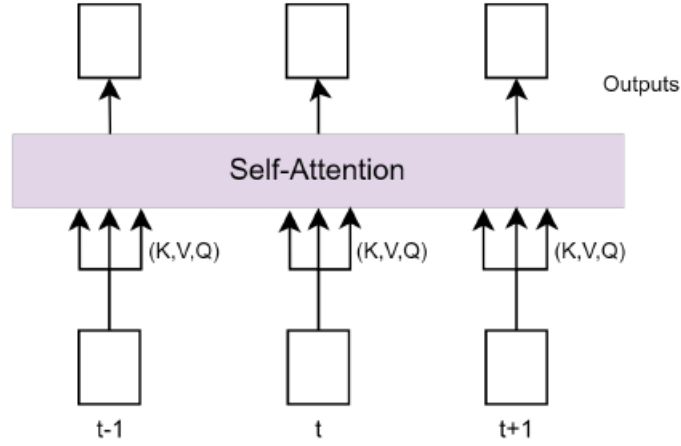


Figure 1.13: Self-attention Architecture [Rahali et al., 2023].

Transformer Advantages for Diacritic Restoration;

Global Context: Dependencies between far-off words in the input sequence can be captured by transformers, which is essential for precise diacritical restoration.

Parallel Computation: Transformers process input sequences more quickly than sequential models like RNNs because of the parallel nature of their attention heads.

Scalability: Unlike RNNs, which are sometimes limited by the possibility of vanishing or bursting gradients, transformers may grow to larger datasets and longer sequences.

1.2.2.4.2. Bidirectional Encoder Representations from Transformer (BERT)

BERT is intended to jointly train on both left and right context in all layers in order to pre-train deep bidirectional representations from unlabeled text. Because of this, without requiring significant changes to the task-specific architecture, the pre-trained BERT model can be improved with just one extra output layer to produce state-of-the-art models for a variety of tasks, including question answering and language inference.

The state-of-the-art language representation model BERT considers the full context of a text sequence, which leads to an improvement over earlier unidirectional models. It makes use of a stack of equivalent Transformer encoders that read the sequence concurrently from left to right and left to right in both directions.

Using a process called transfer learning, BERT trains the model on vast volumes of unlabeled data beforehand, then fine-tuning it on smaller labeled datasets for particular applications. The source model is created with learned parameters during the pre-training phase. Only the output layer parameters—which are learned and supervised from labeled data for the target task—are adjusted during the fine-tuning stage, which begins with the model initialized using the source parameters.

Pre-trained representations, as we demonstrate, eliminate the requirement for numerous intricately designed task-specific structures. First fine-tuning-based representation model, BERT outperforms numerous task-specific architectures and achieves state-of-the-art performance on a wide range of sentence-level and token-level tasks [Devlin et al., 2018].

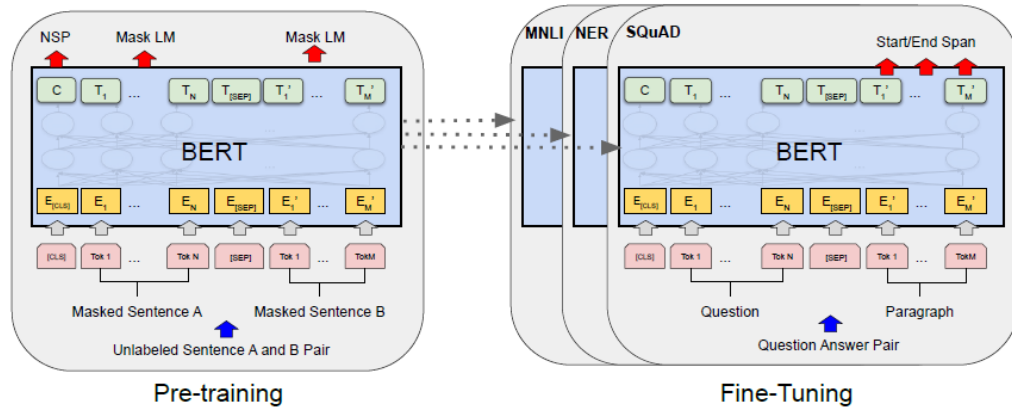


Figure 1.14: Bert Model [Devlin et al., 2018].

For pre-trained BERT, there are two unsupervised learning methods

Masked LM (MLM): To train a deep bidirectional representation, we just need to predict the tokens that have been masked after randomly masking a certain percentage of the input tokens. Although the literature frequently refers to this process as a Cloze task, we refer to it as a masked LM.

Next Sentence Prediction (NSP): NSP is predicated on comprehending the connection between two sentences, which language modeling does not adequately represent. To train a sentence relationship understanding model, we pre-train for a binarized next sentence prediction challenge, which can be easily produced from any monolingual corpus.

Overall, its pre-trained representations allow for effective transfer learning, where less labeled data may still deliver good performance by fine-tuning on certain datasets. BERT's transformer architecture guarantees scalability and interpretability, and its adaptability makes it suitable for multilingual applications. These qualities strengthen BERT's position as a cornerstone of contemporary NLP breakthroughs.

1.2.2.1.4. Generative Pre-trained Transformer (GPT)

OpenAI created the Generative Pre-trained Transformer (GPT) deep learning model, which is intended to comprehend and produce writing that is similar to that of a human. "Improving Language Understanding by Generative Pre-Training" is the paper in which they presented this investigation.

It is predicated on the Transformer architecture, which was presented in Vaswani et al.'s paper "Attention is All You Need" [Vaswani et al., 2017]. GPT models are adjusted for certain tasks after being pre-trained on a big corpus of textual data. The study of Radford et al. shows that significant improvements in these tasks can be achieved by discriminatively fine-tuning a language model after it has undergone generative pre-training on a variety of unlabeled text corpora [Radford et al., 2018].

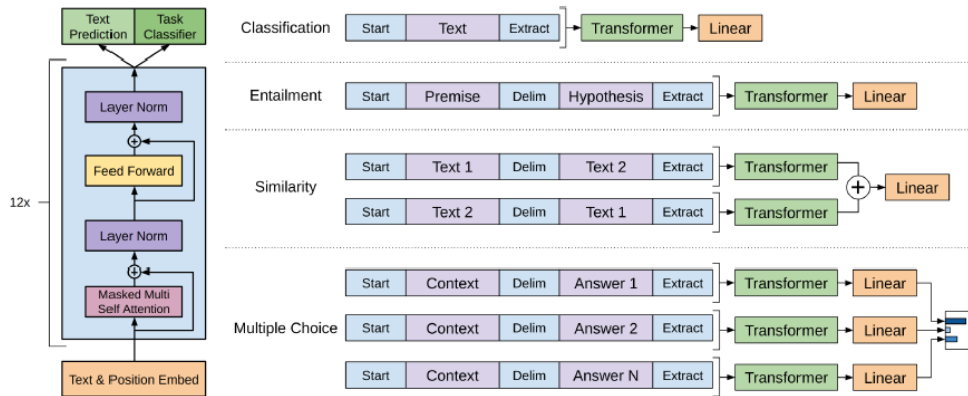


Figure 1.15: GPT Architecture [Radford et al., 2018].

By combining supervised fine-tuning with unsupervised pre-training, they investigate a semi-supervised method for language understanding tasks. Learning a universal representation that can be used to a variety of tasks with minimal modification is their aim. We make the assumption that a sizable corpus of unlabeled text and many datasets containing manually annotated training samples are available (target tasks). The domain in which these target tasks are located does not matter to their configuration in relation to the unlabeled corpus. Radford et al. (2018) use a two-step training process. To train the initial parameters of a neural network model, they first apply a language modeling objective on the unlabeled data. The related supervised goal is then used to adjust these parameters to a target task. The GPT uses only Transformer Decoder with masked self-attention heads.

Radford et al. (2018) look into recent research on semi-supervised learning that shows how employing word embeddings learned on unlabeled corpora can lead to improved performance.

To maximize the likelihood of unsupervised pretraining as the following equation;

$$L_1(U) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \theta) \quad (1.46)$$

In the unsupervised corpus $U = (u_1, \dots, u_n)$, k represents the context window's size, and P denotes the neural network's conditional probability using the parameter θ . To generate an output distribution over target tokens, this model applies position-wise feedforward layers after a multi-headed self-attention operation over the input context tokens:

$$h_0 = UW_e + W_p \quad (1.47)$$

$$h_l = \text{transformer_block}(h_{l-1}) \forall i \in [1, n] \quad (1.48)$$

$$P(u) = \text{softmax}(h_n W_e^T) \quad (1.49)$$

Provide the input parameters $x = x^1, \dots, x^m$ with label y to the pre-trained model for supervised fine-tuning in Eq. 1.48 and maximize in Eq. 1.49. The final transformer model is h_l^m . Linear output layer W_y . C is the labeled dataset and Equation 1.50 is the formulation that results from optimizing the following objective with weight λ .

$$P(y | x^1, \dots, x^m) = \text{softmax}(h_l^m W_y) \quad (1.50)$$

$$L_2(C) = \sum_{(x,y)} \log P(y | x^1, \dots, x^m) \quad (1.51)$$

$$L_3(C) = L_2(C) + \lambda * L_1(C) \quad (1.52)$$

The GPT model stands out due to its high level of popularity. GPT is used for many different types of tasks, including document classification, question-answering, semantic similarity assessment, and textual understanding. The creative Transformer architecture, substantial pre-training on vast datasets, scalability, adaptability through fine-tuning, and capacity to produce human-like language are what give GPT its power. These characteristics make GPT a basis for numerous AI applications and a top tool in the field of natural language processing.

1.2.2.1.5. T5 (Text-to-Text Transfer Transformer)

Raffel et al (2020) who are the Google Researchers created the T5 (Text-to-Text Transfer Transformer) neural network architecture, which is intended to manage a broad range of natural language processing (NLP) activities in a single, cohesive system. The fundamental notion of T5 is to treat all NLP problems as text-to-text tasks, in which text strings are used for both the input and the output. This method makes the architecture simpler and gives task designers greater freedom.

Their work is based on the fundamental principle that all text processing problems can be solved as "text-to-text" problems, meaning that text can be input and fresh text may be output. Previous unifying frameworks for NLP tasks, such as interpreting all text problems as question-answering, language modeling, or span extraction tasks, served as inspiration for this methodology. Most importantly, we can easily apply the same model, goal, training protocol, and decoding method to all tasks we examine using the text-to-text framework [Raffel et al., 2020].

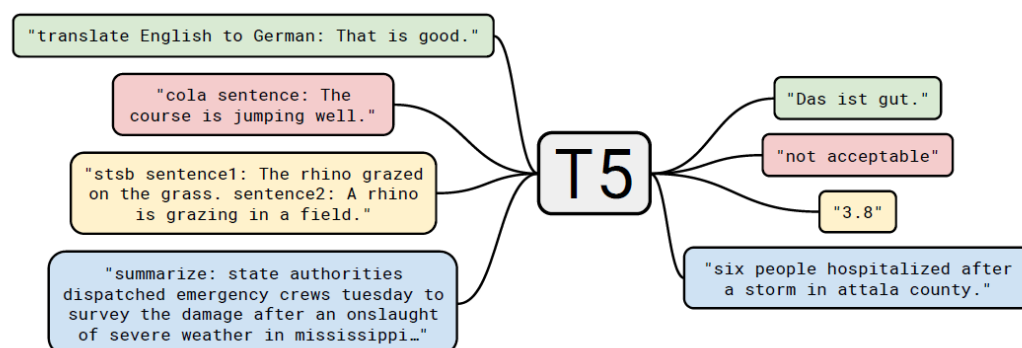


Figure 1.16: Text-to-Text Transfer Transformer Framework [Raffel et al., 2020].

Instead of producing a class index, T5 is taught to produce the exact text of the label (such as "positive" or "negative" for sentiment analysis). This approach's main benefit is that it enables the use of the same training objective (teacher-forced maximum likelihood) for all tasks, therefore allowing for the effective fine-tuning of any downstream task using a single set of hyperparameters. T5 undergoes pre-training on a masked language modeling "span-corruption" objective, in which a mask token is substituted for successive input token spans, and the model is trained to reconstruct the tokens that were masked out. T5 employs a span corruption objective in pre-training, wherein text spans are hidden and the model's job is to rebuild the hidden spans from the surrounding context. This aids in the model's acquisition of the ability to produce text dependent on both local and global context.

T5 is a great option for a variety of NLP tasks due to its strong design, which is based on the Transformer encoder-decoder structure and its adaptable text-to-text framework. There are a lot of fascinating studies using T5. Mastropaol et al.'s study on code-related tasks is one of them. The T5 model was trained for several tasks such as code summarization, assert statement generation in test procedures, automatic bug-fixing, and code injection.

1.2.2.1.6. RoBERTa (Robustly optimized BERT approach)

A version of Facebook AI's BERT (Bidirectional Encoder Representations from Transformers) model is called RoBERTa (Robustly optimized BERT approach) [Liu et al., 2019].

RoBERTa can capture context in both directions (left-to-right and right-to-left) since it employs the same multi-layer bidirectional Transformer encoder design as BERT.

With the help of larger datasets, longer training periods, and a few other performance-enhancing adjustments, it seeks to improve BERT's pre-training procedure.

In comparison to BERT, RoBERTa has been pre-trained on significantly larger datasets. It makes use of more than 160GB of text from datasets such as WebText, the Common Crawl News, and others. The model can view more data and acquire better representations because it is trained for longer periods of time. RoBERTa uses dynamic masking, where the masking pattern varies with each training period, in contrast to BERT's static masking. The model gets exposed to a wider range of

situations as a result.

Removed Next Sentence Prediction (NSP): RoBERTa ignores the NSP goal from BERT in favor of the more successful masked language modeling (MLM) goal.

Larger Batches and Learning Rates: To increase effectiveness and performance, training makes use of larger mini-batches and learning rates.

RoBERTa's strength in natural language processing (NLP) stems from its improved training methods, like dynamic masking and NSP removal, in addition to its capacity to use big datasets and lengthy training periods. It offers state-of-the-art performance and high transfer learning capabilities, making it appropriate for a variety of NLP tasks thanks to its sturdy architecture and adaptable pre-training. When taken as a whole, these characteristics make RoBERTa one of the most potent and successful models in NLP.

1.3. Common Approaches for Diacritic Restoration

Generally, in methods that try to solve the DR problem, sentences taken as input and different structures that provide data input for training are used. In particular, training is provided to correct critical letters in words or characters in the sentence as diacritic characters. In some studies, considering the structure of the language, it is seen that syllable-based studies are also preferred. This decision may vary depending on the structure of the language being studied and the structure of the preferred model.

1.3.1. Word-Based Approach

A word is viewed as a fundamental unit of information that is subsequently processed in the majority of diacritics restoration studies. Other methods, many of which rely on dictionaries and select words from a list of ambiguous words depending on the context provided by neighboring words, have also been proposed and shown to be successful [Suppa, 2018].

Based on a review of the literature, most tokens (or words, in this case) that are taken into consideration for the restoration of word-based diacritics have only one change between the accented and unaccented forms of the token. Put otherwise, the majority of the time there is no accent to save. Furthermore, one repairable word is typically used more frequently and is consequently regarded as dominant when there is

ambiguity between them. This is true for all languages, not only European ones like French and Spanish [Yarowsky, 1999]. Even with its simplicity, this principle, when applied to a generic measure of accuracy, can yield quite robust findings. The authors of [Yarowsky, 1999] state that the average accuracy for French is 97.6%, and the average accuracy for Spanish is 98.7%.

Looking at these results, we can still conclude that there are a lot of errors in the accuracy rate and that the optimum approach is not word-based. We realize that the ambiguous words are the primary concern here. For higher success rates, this is where the attention must be directed.

1.3.2. Character-Based Approach

In general, word-based accent restoration is not thought to be a very tough operation, and even with pretty simple models, it produces quite decent results.

But as Suppa M. (2018) points out, the majority of these techniques fall short in the following situations:

1. There are no electronic dictionaries accessible for the target language or the public library only has small-print dictionaries. When diacritical marks are absent from the dictionary, a specific circumstance occurs; in this instance, the dictionary's value for correcting accents is severely restricted.
2. There are no publicly accessible morphological and/or syntactic analysis tools for the target language (the Part of Speech tagger).
3. The dataset used to train the accent restoration model is not very large.

A crucial path in the restoration of diacritics is character-level methods. They resolve issues that are commonly categorized as labeling sequences. By assigning their diacritical equivalents (labels), an iterative process traverses an undiacritical sequence of characters. The surrounding material acts as additional classification features and each character is handled as a distinct categorization instance. Such approaches are appropriate for languages with fewer resources because they typically only require the raw text and no further language tools. Character-level approaches are also resilient to unfamiliar terms. This classification process can be interpreted as either the sequence classification problem (which takes conditional dependencies between predictions) or

the independent instance-based classification problem (which assumes that each instance is independent) depending on the classifier selected [Stankevičius, 2022].

1.3.3. Syllable-Based Approach

In general, the diacritic restoration problem is solved more frequently using word- and character-based techniques. Studies have been conducted on this, however, it has been noted in some that the linguistic structure of some languages—like Turkish—is better suited for syllabification and that the results of the syllable-based analysis are intriguing. It has also been noted that the linguistic structure and data employed in certain research have an impact on the outcomes, either successful or unsuccessful

In Nguyen et al.'s character- and syllable-based analysis, for instance, it was shown that the character-based structure produced superior results [Nguyen et al., 2010]. Nevertheless, it has been noted that studies focusing on syllables have been conducted for languages with appropriate grammatical structures, like Vietnamese, Arabic, and Thai. To our knowledge, no such study has been conducted for Turkish yet.

1.4. Evaluation Metrics

Model generalization in Natural Language Processing (NLP) classification tasks like sentiment analysis and topic categorization are typically assessed using common metrics like accuracy, precision, recall, F1 score, and AUC-ROC.

Table 1.6: Confusion Matrix

Actual \ Predicted	Positive Prediction (P)	Negative Prediction (N)
Positive (P)	True Positive (TP)	False Negative (FN)
Negative (N)	False Positive (FP)	True Negative (TN)

1.4.1. Accuracy

The proportion of correctly identified samples to total samples is known as accuracy. When all classes have an equal number in a balanced data collection, it can be a useful performance indicator.

The heuristic that the optimal model will have the most properly predicted instances is encoded by accuracy. This prior permits the "accuracy paradox," in which a naive model could incorrectly predict the most prevalent class, erroneously boosting the generalizability score [Vickers et al., 2024].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1.53)$$

$$Accuracy = \frac{1}{S} \sum_{c=0}^C TP_c \quad (1.54)$$

1.4.2. Precision

The proportion of accurately predicted positive observations to all predicted positive observations.

$$Precision = \frac{TP}{TP + FP} \quad (1.55)$$

1.4.3. Recall

The proportion of all observations made in the actual class to all positively predicted observations.

$$Recall = \frac{TP}{TP + FN} \quad (1.56)$$

1.4.4. F1 Score

The geometric mean of a binary classifier's precision and recall is the definition of this statistic.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (1.57)$$

$$F1\ Score = \frac{1}{C} \sum_{c=0}^C \frac{TP_c}{TP_c + \frac{1}{2}(FP_c + FN_c)} \quad (1.58)$$

In general, the F1 score is thought to be a far more reliable indicator of relevance than accuracy since it considers more than just a model's errors. It is more logical to utilize F1 score rather than accuracy when the context is dynamic.

The authors in Ungurean et al. (2008) employed these metrics, particularly the F1 score, to assess their unigram, bigram, and trigram models, although in a different setting (character level diacritics restoration).

2. PURPOSE OF THE STUDY

Diacritics are essential to meaning in Turkish. Misunderstandings may arise from the misuse or omission of these marks. Regrettably, in digital texts diacritical markings are commonly missed, requiring efficient restoration techniques.

The majority of current techniques for fixing Turkish accents are character- or word-based, and they might not fully capture the subtleties of the language's syllable structure. Research on syllable-based diacritic restoration strategies is noticeably lacking, despite the possibility of more precise and context-sensitive findings.

This study's primary goal is to create and assess a syllable-based method for correcting accents in Turkish literature. Particular goals are:

- To discover a useful and simple labeling technique due to its suitability to the syllabic structure of Turkish. To establish a model for diacritic correction improvement utilizing syllable-based data.
- To evaluate the syllable-based approach's performance against the current word- and character-based techniques.
- To demonstrate the applicability of the suggested approach to languages with different syllabic structures.

A method for accent correction that is more linguistically aware and based on syllables may yield higher accuracy and improved contextual comprehension, since it is difficult to teach every word to the model and an infinite variety of terms is needed for word-based learning. Conversely, character-based approaches lack semantic depth or are meaning-insensitive. The advantage of using syllables is that the number of syllables in Turkish is much less compared to the number of words. The specifics of why syllable-based learning is appropriate for Turkish are covered in section 2.1. Consequently, it is believed that the learning ability of the model will increase and the probability of making errors will decrease.

This study uses the Turkish language's natural language units to fill the gap in current techniques. By offering improved Turkish accent correction, the study's findings can enhance text processing software and natural language processing (NLP) tools. They can also feed into future research endeavors.

The written Turkish texts that we shall examine in this research are sourced from Wikipedia. A denoised corpus created especially for this project will be used for model development and testing.

The study aims to demonstrate that, in terms of accuracy and contextual relevance, a syllable-based approach to accent correction can perform better than current character- and word-based techniques. It is anticipated that the results will aid in the creation of more sophisticated natural language processing (NLP) systems for Turkish and other languages with comparable spelling challenges.

2.1. Syllable-Based Approach for Turkish Diacritic Restoration

Why is it logical to use a syllable-based system? Since it is simpler for the transformer to determine the order in which a sufficient number of significant entities are related to one another. Text structures that are transformed into letter-based entities have vocabularies that are about equal to the total amount of letters in the language plus a few special characters. It will help the transformer understand the connections between a small number of entities if it has a small vocabulary. Nonetheless, learning the relationships between individual letters does not always lead to effective learning, as letters by themselves do not convey meaning. Numerous studies have demonstrated that letter-based approaches, particularly for measuring semantic success, are not very effective. In contrast, words are meaningful units on their own. Therefore, it can be argued that a word-based data model would enhance semantic learning. However, word-based structures also have their drawbacks. Vocabularies in word-based applications are larger than hundreds of thousands. The linguistic structure of some languages, like Turkish, allows for the derivation of hundreds of words from a single root word. This indicates that there are millions of words in the Turkish language and that its size is uncontrollable. Enough training data cannot be found to learn every word and all of its statistical correlations. But when syllable-based data models are taken into account, the majority of Turkish syllables are made up of one, two, three, or four letters. In order to form a syllable, these potential variations also adhere to very rigorous guidelines. This indicates that it is a syllable vocabulary with a possible 2500 entities, which is a small and acceptable quantity. Just a few thousand words...

Nevertheless, in Turkish, syllables are typically significant textual units. Furthermore, there are restrictions that a word's successive syllables must obey. These are crucial in helping the transformer understand the context of the diacritical correction issue. Thus, a structure based on syllables would improve the problem's semantic success rate. This led to the creation of a syllable-based data model, which is a crucial component of the technique.

2.2. Sequence to Sequence Transformer Model

Transformer models, which provide state-of-the-art performance in many applications, have gained particular attention recently for diacritic restoration due to their capacity to understand context and manage long sequences effectively. In sections 1.2.2.3 and 1.2.3.4, it was covered in detail. So we also preferred the use Transformer Model in this study.

Sequence-to-sequence Transformers Machine translation is typically well executed by transformers. Translation between languages is possible using it. In addition, it can be applied to tasks like text summarizing and answering questions. Naturally, it can also offer a workable answer to the issue we raise in this study.

A sequence-to-sequence Transformer model that was trained on the job of machine transformation from diacritically inaccurate text to ideal text was preferred in this investigation. The encoder and decoder that make up the suggested sequence-to-sequence transformer are connected. Additionally, the model was taught about syllable order through the use of the Positional Embedding layer. This architecture transfers the source entity sequence—which could contain diacritical errors—to the encoder structure, which creates a new representation of the sequence. The transformer decoder will then import this updated representation. Next, the transformer decoder will attempt to predict the target sequence's entities.

The syllable-based sequencing dataset was produced. In addition to syllables, other items with meaning in sequence data are included in the vocabulary that contains these syllables. These significant elements were arranged in the syllable-based sequential text data in the proper sequence. The upcoming sections 3.2 and 3.3 will provide a detailed explanation of these two preprocesses. Next, to train and operate the transformer, the sequence text data were transformed into integer arrays in the form of

vectors, where each vector represents an entity in the vocabulary. This procedure, known as vectorization, was carried out to operate and train the transformer.

An encoder and a decoder make up a sequence-to-sequence structure. We used inputs, sample outputs, and outputs that we supplied data as the source, target, and prediction, respectively, when training the transformer. To finish the training, we employed a pair structure in Figure 2.1 as {(inputs: source), (sample outputs: target)} => {outputs -> prediction}. A comparison was made between the target and predicted outcomes.

Source: English equivalent characters for diacritical marks in syllabled sentences

Target: Syllable-based sentences with their proper forms in Turkish

Prediction: Turkish prediction forms for a syllabled sentence.

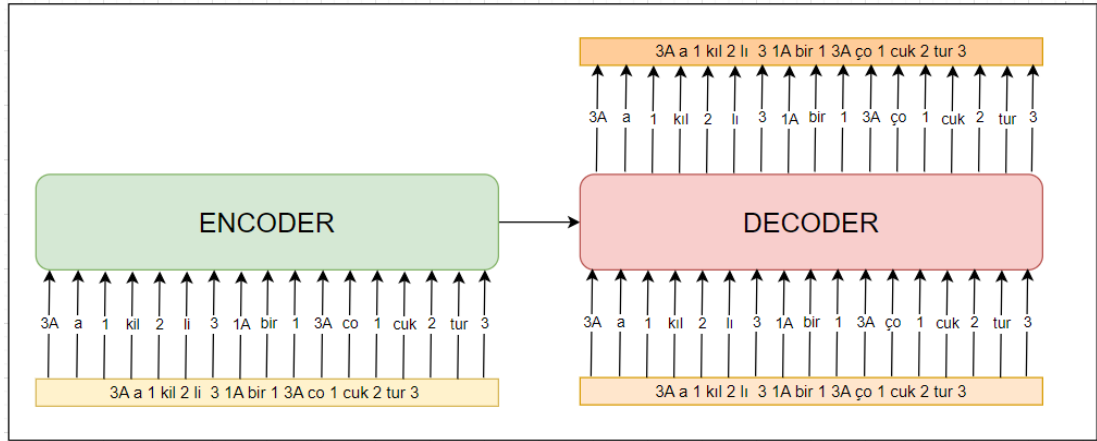


Figure 2.1: Syllable Sequential Labeling Transformer Encoder-Decoder Model.

Seq2Seq transformers are especially effective for diacritic restoration because of their rich, context-aware embeddings from pre-trained models, their efficiency and scalability from parallel processing, and their capacity to capture long-range dependencies and contextual relationships through self-attention. Because of these characteristics, Seq2Seq transformers are a perfect fit for this purpose since they can reliably forecast the proper placement of diacritical restoration and comprehend the context in which words are employed.

3. EXPERIMENT

This section begins with a discussion of our model's overall structure in section 3.1. After that, topics related to data modification and corpus will be covered in section 3.2 and 3.3. Four different experiments will next be examined in section 3.4.

3.1. Model

The function that converts a text with some characters that require diacritical correction into an ideal text can be characterized as the study's problem. The fundamental subcomponent function of this primary function is the trained transformer network structure that should be suggested. First, the input and output data models are used to train the transformer model. Instead of being raw texts, input and output data models are preprocessed text pairs. The input data model represents raw input text data with diacritic faults using syllables. The perfect raw output text data without any flaws is represented by the output data model. Some symmetrical, meaningful sequence items are also present in the input and output of these data models.

There are three fundamental steps in the design of training data. To reduce noise and facilitate learning, expressions that deviate from the main focus of the problem are first simplified in the data. After that, the data is produced as syllable-based data. Lastly, the data model is expanded with sequentially meaningful entities that describe restrictions and rules. Models of the intended input and output data are used to train the transformer model. Later on in the learning process, the trained model is preserved as a subfunction of the issue (Figure 3.1).

The transformer is optimized between the sequential input data model and the output data model during the training phase. The primary aspect of network model training is the sequence statistics between these textual sequence entities. Vocabulary is this set that includes every entity in the text. Another crucial factor in the transformer learning process is vocabulary size. The transformer's learning process involves determining which entity, with what statistics and probability, could be located after

various entity sequences. As a result, there must be enough sequential data instances containing each thing in the vocabulary for it to function better.

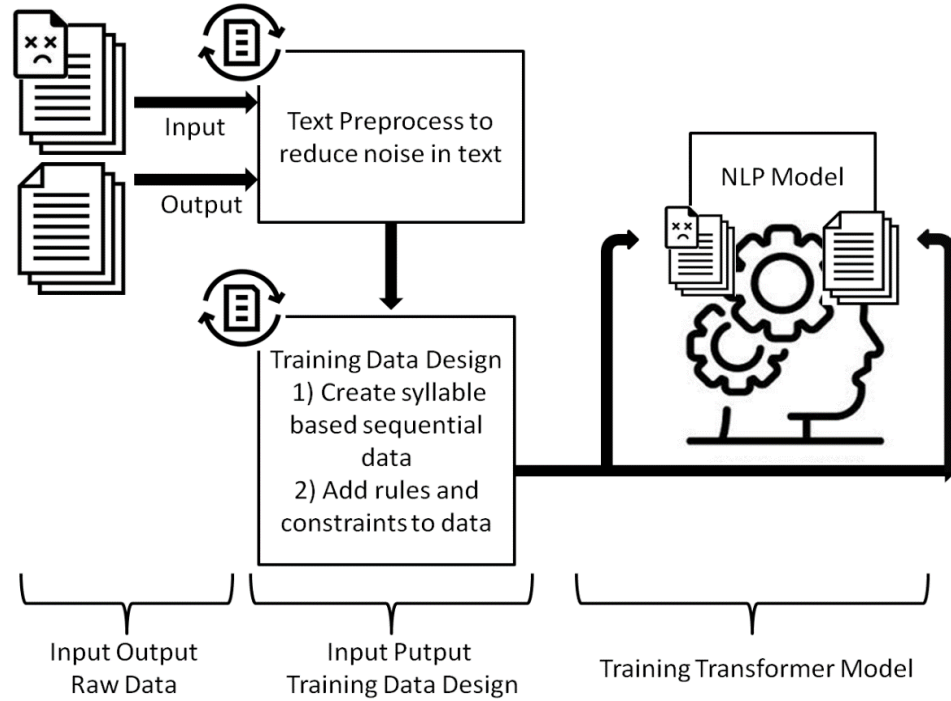


Figure 3.1: The Overall Study.

By incorporating more semantic information into the syllable-based data, the data can be further enhanced. The data may use embedded information such as the number of syllables in a word and the syllable order within a word. Nonetheless, the data's inherent statistical structure ought to be maintained during the data enrichment process. This raw data's statistical structure will be upset if this information is loaded into the system by directly altering the vocabulary's entities. Consequently, by inserting the required labels between these syllables in a specific way and sequentially, the data can be enhanced with some semantic information without altering the order between the text's real syllables. In section 3.4.4, this technique—also referred to as sequential labeling of data—is covered in more detail.

Within the framework of the problem's content, several language contents don't particularly stand out as structures. Date expressions containing significant years of wars, revolutions, etc. are one example. The book contains thousands of year

expressions as numbers. It is common for Turkish corpora to have syllable counts of little more than 4,000. Even so, texts containing merely the dates of historical events can have numbers in the thousands. Expressing this text content that is not at all related to the problem's focal point. For the transformer, thousands of items in the language produce a lot of noise. One of the components of this study, which is covered in detail in section 3.3, is preventing this and similar noises for the transformer to succeed.

3.2. Corpus

A sizable corpus of Turkish texts from Wikipedia served as the study's training, test, and validation data sources. The data was initially cleaned up to remove non-Turkish words and make it appropriate for analysis. Subsequently, the raw data underwent a few preparation processes that are detailed in detail under heading 3.3 in the method sections.

In particular, the state-of-art model that was found after Experiment 4. Thus, those entities were specifically computed for "Syllable Sequential Labeling," or Experiment 4, as it is known.

The corpus used contains 473,412 sample sentence pairs. This dataset has 5,761,450 words in it. There are 36,859,891 entities in the syllable-based sequence data that was designed. 21,073,965 of these entities are symbolic entities that are used to introduce rules and limitations, whereas 15,785,926 of them are true syllabic things. Each sentence's input/output pair was made in the manner depicted in Figure 3.1 and described in the approach section. 92% of the data were utilized for training, 4% for validation, and 4% for testing.

There are 2049 entities in the syllable vocabulary that match the complete input/output pair data. Furthermore, since Turkish characters are absent from the transformer's input data, the syllable vocabulary that matches the input data consists of 1552 entities.

These input/output data pairings, which were intended to be as straightforward and limited in language as feasible, were used to train the transformer.

3.3. Data Preparation

Maintaining textual integrity while correctly constructing syllables is crucial for resolving the diacritic problem in the study. In the process, the transformer will function better if a vocabulary comprised of all the various syllables is created in a way that is inclusive and has fewer parts. Consequently, it would be advantageous to employ specific symbols for structures that convey the same semantic information. Eliminating textual structures that aren't necessary to solve the problem may also be beneficial. To guarantee the input/output data model's simplicity in this situation, the auxiliary techniques listed below were applied. The morphologic analyzer tool was also used to exclude foreign and rare words from the text in addition to the actions listed below. Naturally, the transformer's performance improved after all of these procedures.

In this case, noise was decreased by performing data cleaning and processing procedures. The processes carried out using the cleaned data would yield more trustworthy results. Below is a detailed explanation of the step-by-step procedure.

3.3.1. Removing Punctuation Marks

There isn't much content information about the relationships between syllables contained in punctuation marks. Because of this, punctuation marks were regarded as nonexistent in this study and were examined by deleting them from the data models. The transformer was specifically designed to concentrate on the link between syllables throughout the training phase. Punctuation markings have been shown to have semantic connotations in certain investigations. As a result, it has been noted that adding punctuation to a vocabulary improves network efficiency. Nonetheless, the majority of the studies referenced are word-based. We eliminated punctuation from the data because we did not observe this beneficial effect in the experiments conducted for this investigation. The only punctuation marks that have been substituted with the "." character are those that denote the end of sentences, such as "?", "!", etc.

3.3.2. Replacing Symbols with Their Word Equivalents

The corresponding words were used in place of several characters. For instance, "percent" is shown by the symbol "%". "Yüzde" is its equivalent in Turkish. "Yüz de"

is its corresponding syllabary structure. Thus, data based on syllables was enhanced. The data structure's simplicity was guaranteed at the same time. Other instances include £="av ro" and \$ = "do lar".

3.3.3. Use Only Lower-Case Characters

Even if part of the letters in a syllable are written in uppercase or lowercase throughout the text, the syllable will still be expressed in the same way. For instance, there are four ways to write the word "su," which means "water." "SU," "Su," "su," and "sU." It is highly likely that data will have at least the first three of these kinds. There is no semantic distinction between them. As such, it makes perfect sense to treat these semantically equivalent structures as a single entity. Because of this, all data was changed to lowercase and lowercase text was established as the standard. Consequently, an expanding vocabulary was stopped. Simultaneously, this prevented the rare entity problem that could arise from the probability of certain syllable forms occurring very infrequently.

3.3.4. Representation of Numbers with Common Symbols

Particularly, digit characters are used to write the text's numbers. There's no way to spell these numbers. Moreover, a corpus contains thousands of distinct numbers. The transformer is hard to master since each integer has its own entity vector. Alternatively, a collection of symbol entities can be used to represent all integers.

In the present study, each digit expression was replaced with the character "N". Nonetheless, the final digit of the number was represented by the symbols "NX", "NK", and "NZ". The "NX" symbol was utilized if the last syllable of this number featured a front vowel, while the "NK" symbol was used in its place if it contained a rear vowel. The symbol "NZ" is used if the digit ends in "0". The vowel character of the final syllable determines whether front or back vowel letters are used to produce suffixes in Turkish.

Therefore, the suggested technique allows the network to learn the adding suffix process. because the network is given the phonetic character of the last syllable of the number, whether it is a front or back vowel. "1993'ün," for example, is a numerical expression. In the training data, this integer takes the form "NNNNX un" for input and "NNNNX ün" for output. It is evident that NNNNX or NNNNK is an easy way to

express a lot of four-digit numbers. This method drastically cuts down on the vocabulary's element count.

The word "5." is used in the sense of fifth in certain texts. "Beş" is the Turkish word for five. The word "beşinci" is the equivalent of the "5." However, suffixes like "-inci", "-üncü", "-nci", "ıncı", "-ncı", "uncu", etc. are used for distinct numbers. The nature of the following suffixes is determined by the variations in these usage. Consequently, these numbers were displayed with specific symbols in writings that used dot characters. If the number finishes with one of the characters "1., 2., 3., 4., 5., 7., or 8," the "NXT" symbol was utilized. For some, the symbol "NKT" was used for "6., 9.," and for others, "NZT" was used for "0." Expressions like "12.", "13.", and "48." for instance, would display the sign "NNXT". This implies that several numbers can be substituted with a single vector. Additionally maintained is the correspondence between consecutive syllables with back vowels and those without.

3.4. Experiments

As we move through our studies, we shall delve into a discussion of the four distinct methodologies that have been developed in the course of our experiments, addressing each one individually.

3.4.1. Experiment-1: Syllable-Based Training with Semantic Makers

Initially, we identified a challenge that resulted from the inability of the syllables in our data set to be combined consistently to form words. The trained model exhibited difficulty distinguishing word boundaries and frequently attempted to produce words lacking the predetermined syllable structure.

As a result, we proposed the use of a character as a delimiter to serve as a word separator. This approach was designed to facilitate the model's comprehension of word boundaries, both at the beginning and end of words. Upon implementation, we achieved an impressive success rate exceeding 96% over multiple training epochs.

We manipulated source sentences such that the sentence using English equivalent characters for diacritic in a syllabled sentence as in (1), the Turkish correct form of a syllabled sentence of which is (2), and also the English meaning is (3).

“Bang la des dun ya nin nu fus yo gun lu gu ba ki min dan en ka la ba lik ul ke le rin den dir .” (1)

“Bang la deş dün ya nın nü fus yo ğun lu ğu ba kı min dan en ka la ba lık ül ke le rin den dir.” (2)

“Bangladesh is one of the most populous countries in the world in terms of population density.” (3)

After using semantic maker the sentence is separated by a word separator.

“Bang la des _ dun ya nin _ nu fus _ yo gun lu gu _ ba ki min dan _ en _ ka la ba lik _ ul ke le rin den dir _ .”

Syllable-Based Training (Syl-B): To delve further into our methodology, we segmented sentences into syllables and conducted transformer training using input and output textual data. This approach yielded a commendable success rate of 94.33%.

Syllable-Based Training with Semantic Makers (Syl-B + Exp1): Additionally, we conducted specific training to instruct the model on word endings, resulting in a notably improved success rate of 96.27%.

Character-Based Evaluation (Char-B): When evaluating the model's performance in a character-based context, we observed a success rate of 85.87%. Nonetheless, it became evident that the syllable-based approach consistently yielded superior results, prompting us to pursue this avenue of investigation.

The specific results are displayed in Table 3.1, where it is evident that the best results were obtained with the discriminative method known as "Semantic maker." More clarification is needed to elaborate on the characteristics and importance of this "Semantic maker".

In the first experiment, it was observed that the use of semantic markers to distinguish boundaries between words and syllables within sentences organized as syllable sequences greatly enhanced the model's training performance. This method was presented as a revolutionary methodology, called "Syllable and Indexing Labeling," which added a fresh perspective to our research.

It was observed that the model encountered difficulties in correctly interpreting certain Turkish language constructs, such as the words "de" and "da," which are contextually equivalent to "dahi" The model often misidentified these words, leading to erroneous

word segmentation, wherein words were incorrectly separated from their constituent syllables.

Table 3.1: The result of Experiment 1.

Experiment 1	Accuracy %
Char-B	85.87
Syl-B	94.33
Syl-B + Exp-1	96.27

Table 3.1's results helped us understand why employing a semantic maker resulted in more training on the transformer. Therefore, we decided to enhance the semantic maker's utilization in the upcoming tests.

3.4.2. Experiment-2: Total Number of Syllables Indexed Labeling

To resolve the problem that arose in the first experiment—namely, the difficulty in distinguishing between "de" and "da"—a new labeling strategy was implemented. In particular, terms like "de" and "da" were classified as "de1," and other words were labeled according to their syllable counts when they were used to express the meaning of "dahi" (meaning "also" or "too"). This modification successfully lessened the issue, which enhanced the model's training. Across epochs, the success rate was more than 98%.

In the example, the sentence's labeling is indexed by the total number of syllables indexed labeling.

“Bang3 la3 des3 _ dun3 ya3 nin3 _ nu2 fus2 _ yo4 gun4 lu4 gu4 _ ba4 ki4 min4 dan4 _ en1 _ ka4 la4 ba4 lik4 _ ul6 ke6 le6 rin6 den6 dir6 _ .”

Nevertheless, a new problem surfaced throughout this process: the model incorrectly recognized "de1," "de2," "de3," and so on as separate words even though they had the same meaning. As such, the principal disadvantage of this methodology is the proliferation of different syllable iterations, which causes the vector space to grow. From 3500 at the beginning, there are now more than 7500 different sorts of syllables. Even while "de2" and "de3" are syllables that have the same meaning, they are

represented by different vectors, which may limit the model's ability to learn the word "de" in a more complete manner.

3.4.3. Experiment-3: The Syllables Indexed Sequential Labeling

We suggested a different method where we directly labeled the syllables with their corresponding syllable numbers, as opposed to labeling terms based on the number of syllables in the word they are contained in. This distinction is especially important in Turkish because of a fundamental feature of the language: a word's root construction is usually represented by its initial one, two, or three syllables, while inflections are frequently shown by the subsequent syllables. We sought to promote a more nuanced interpretation by clearly marking syllables as 1 and 2, acknowledging that these first syllables may have different meanings and linguistic purposes.

We labeled indexed sequentially for each syllable in the example sentence.

“Bang1 la2 des3 _ dun1 ya2 nin3 _ nu1 fus2 _ yo1 gun2 lu3 gu4 _ ba1 ki2 min3 dan4 _ en1 _ ka1 la2 ba3 lik4 _ ul1 ke2 le3 rin4 den5 dir6 _.”

Notably, the labeling of syllables 1 and 2 introduces an intriguing challenge, where multiple interpretations can arise. For instance, in the context of the sentence "raf a ra sin da k1 rik cam var," signifying "there is broken glass between the shelf," the model erroneously associates the sequence "arasında..." with the necessity of appending "ki." Subsequent errors in sentence transformation result from this initial misinterpretation.

To address this issue, we propose the systematic labeling of the syllable "k1" in the term "k1 rik" as "ki1" to distinguish it from the inflectional suffix "ki6." This labeling approach minimizes ambiguity and enhances accuracy without necessitating a significant expansion of the syllable dictionaries. To streamline this process, a common tag is employed for other syllable orders.

It is noteworthy that phrases in syllables 3 and beyond may convey equivalent meanings. During the training process, we achieved an epoch success rate of approximately 99.05%. However, the challenge of syllable lexicon expansion persists and remains considerable. This expansion has led to increased fragmentation, particularly affecting rare syllables. The scarcity of syllable occurrences constitutes the primary factor hindering the model's learning.

The order in which these syllables are learned is somewhat overlearned, but the underlying problems of vocabulary expansion and inconsistent categorization of syllables with comparable meanings are still unresolved. Our considerable experimentation has led us to develop the following guiding principles:

- **Minimize Syllabic Complexity:** To facilitate model comprehension, reduce the overall complexity of syllables.
- **Improve Model Understanding:** To help the model better comprehend Turkish morphology, explicit word structure tagging is introduced.
- **Handle Ambiguity in Syllable Order:** Address the issue of unclear syllable order by implementing uniform labeling procedures for syllables that have similar meanings.

These principles provide valuable insights into the enhancement of Turkish language processing models, with potential relevance to the broader field of natural language processing research.

3.4.4. Experiment-4: The State of Art Model Syllable Sequential Labeling

Certain guidelines are followed while entering a sentence into the model:

- **Specification of Syllable Count:** The model is provided with information regarding the number of syllables present in each word within the sentence.
- **Syllable Sequence Indication:** The order of syllables in the sentence is explicitly conveyed to the model, ensuring accurate interpretation.
- **Minimization of Syllable Lexicon Expansion:** It is a principle to avoid unnecessary expansion of the syllable dictionary during the aforementioned steps.

These guidelines are observed to maintain the integrity of the model's understanding and processing of linguistic input.

As evident in the provided methodology, a sequential tagging system using labels such as 1A, 2A, 3A, and so forth has been incorporated seamlessly alongside each word, without disrupting the inherent syllable structure. By integrating these tags within the syllabic sequence, the model gains a comprehensive understanding of the syllable

count following each tag, thereby preserving the original syllabic structure of the words.

Furthermore, when the model encounters a sequence like "...«bu»->"1"...", it promptly recognizes that "bu" represents the first syllable, facilitating the accurate identification of "bu" as the demonstrative adjective. This approach ensures the preservation of the input word structure, resulting in output sequences that maintain the same structure. For instance, if the input structure were presented as "...3A syllable1 1 syllable2 2 syllable3 3...", the corresponding output structure would mirror it as "...3A Prediction1 1 Prediction2 2 Prediction3 3...".

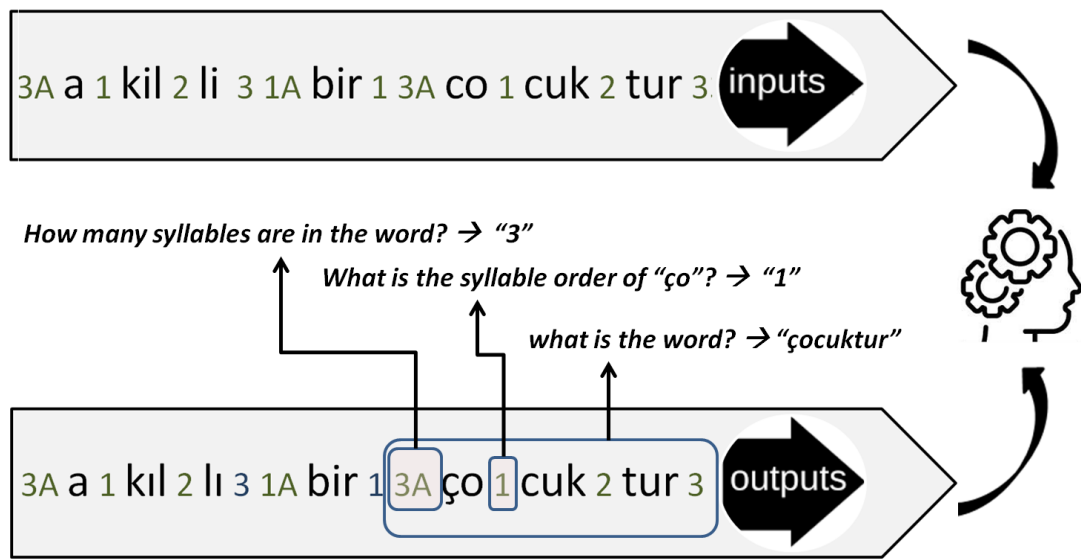


Figure 3.2: The Sequential Labeling.

We can also show an example sentence for the Syllable Tagging of the Sequential Labeling on the transformer encoder-decoder model. In the below example, the sentence using English equivalent characters for diacritic in syllabled sentences as in (1), the Turkish correct form syllabled sentences (2) are shown in Figure 3.2. Context window size n is shown as the number of syllables including taggings. $n = 17$ is the example sentence. We notified the maximum number of context window sizes can be 60 in our model.

Figure 2 shows the input-output training data pair instance for a short sentence. The ideal sentence consists of 3 words: "akıllı", "bir" and "çocuktu". The raw syllable-base output sequence for this sentence is: "a kıl lı bir ç o cuk tur". The first word of the sentence, "akıllı", means "smart". However, when we consider the phrase "kılı",

which is part of the first word, it refers to a word meaning “hairy”. A diacritic combination of this word is the meaningful word "killi". This word means “clayey”. At the same time, the syllables "kil" and "kıl" are meaningful words on their own. All these cases present a possible input-output relationship for the trained transformer function. The transformer function, whose parameters are optimized according to the training data, selects the best among them for output. Most of the time, word orders with relatively less usage will cause some syllables to appear together more frequently or not in the training data. This can often cause the trained transformer to choose the wrong one among the possible output possibilities. Eliminating possible wrong options allows the transformer to make more accurate decisions within the transformer's functional logic. The sequential labeling method comes into play in these cases. For example, imagine that we transformed the word "akıllı" as in Figure 2: for input, "3A a 1 kil 2 li 3", for output, "3A a 1 kıl 2 lı 3", and imagine that we applied this method to all the data. Entities such as "1A, 2A, 3A" are expressions that indicate how many syllables each word consists of. Expressions such as "1, 2, 3" that come after the syllables indicate the syllable order of the preceding syllable in the word. These entities will be the most frequently occurring expressions in the sequential data. Naturally, the meaning of these expressions will be analyzed very well by the transformer. When a trained transformer sees the expression "3A", it will know that statistically it will be followed by something like "3A → syl1 1 syl2 2 syl3 3". If a syllable does not contain a diacritic character, it is not changed at all. Transformer learned this situation from thousands of lines of data. Therefore, if the input started as "3A a", it is easily created as "3A a" in the output. There are 2 diacritic characters in the following 2 syllables and there are 4 possibilities for output. Among these, only the option "3A a 1 kıl 2 lı 3" expresses a meaningful word. The options “3A a 1 kil 2 li 3”, “3A a 1 kil 2 lı 3”, and “3A a 1 kıl 2 li 3” are not meaningful sequences. They probably never occurred in the data. Naturally, the transformer will easily find the correct option as output.

When considering such examples, the trained transformer makes decisions based on the statistical relationship between syllables within this structure and also learns the relationship between words. Because both syllable-based sequence data were used and words information was used as sequence pattern. Transformer model learns the relationships between syllables of the same word. At the same time, It makes sense of separately the relationships between syllables of different words..

The main value of this method part is that it prevents the loss of content information that occurs when switching from a word-based system to a syllable-based system. In addition, it is the transfer of information that can be extracted from the word-based system to the syllable-based system. In this switching phase, also syllable information is added to the data with a pre-process. This preprocessing is an expert function and enriches the data in terms of content. However, the word content is lost. Even so, it is possible to reload this content into the data with another preprocess. This was implemented with the sequential labeling method described above and made a great contribution to the success of the problem

4. RESULTS

The method used in this study first converts the input data completely into English letters. This principle is common in the training process and the prediction process with trained transformers. Therefore, regardless of the amount of diacritic distortion in a given text, the input data will be the same. All syllables containing critical letters carry the potential for change. Expressing these with only English characters provides simplicity in the input section. This can be thought of as encoding. The transformer is expected to predict the correct diacritic spelling from this input structure. In the field of machine learning, many different criteria are used to measure prediction success. Criteria such as recall and F1 score are widely used. These criteria are important in some cases. For example, in the problem of matching a vector with one of two classes, if the percentage of a class in the data studied is very low. This percentage will affect the value of the recall and F1 score. However, our method technique is independent of the corruption of the data as mentioned above. The transformer is trained regardless of the amount of distortion in the input dataset, and its learning is not affected by this amount. Since the text data used in different studies on this subject have different rates of distortion, it is not reasonable to compare different studies with these criteria. Therefore, the accuracy rate was used as the success parameter.

Initially, we will examine the total experiment findings for results based on syllables.

This is explained in section 4.1.

We named the state-of-the-art model Syllable Sequential Tagging in Experiment 4, and we'll be analyzing more of its findings in section 4.2. Its success, especially with ambiguous words, is one of the main goals of this study. Here, using the morphological analyzer, it can be tested whether any word is valid for any language. Often a meaningful word exists for only one variation of the critical letters in a word. What is important in terms of semantics is to make the right choice among ambiguous words that appear in many different variations. Thus, it was observed that this labeling method achieved successful results on ambiguous words. That's why we want to compare the results with different studies. In this context, the prediction success of ambiguous words is given and discussed in 4.2.3. Apart from this, the prediction

success of critical letters and the correct prediction success of words containing critical letters were determined for comparison purposes and are presented in 4.2.1 and 4.2.2.

4.1. Overall Experiments Syllable-Based Results

In this section, Table 4.1 displays the unsuccessful characters in red bold that indicate wrong prediction by Transformer model, while only the successful portions of the transformer's source (input), target, and prediction (output) samples are shown in black bold.

Table 4.1: The result of an example set in diacritic correction.

Source	2A ku 1 ru 2 1A col 1 3A ik 1 li 2 mi 3 .
Target	2A ku 1 ru 2 1A çöl 1 3A ik 1 li 2 mi 3 .
Prediction	2a ku 1 ru 2 1a çöl 1 3a ik 1 li 2 mi 3 .
Source	2A gu 1 ney 2 2A yu 1 zu 2 1A dik 1 2A ku 1 zey 2 2A yu 1 zu 2 4A me 1 yıl 2 li 3 dir 4 .
Target	2A gü 1 ney 2 2A yü 1 zü 2 1A dik 1 2A ku 1 zey 2 2A yü 1 zü 2 4A me 1 yıl 2 li 3 dir 4 .
Prediction	2a gü 1 ney 2 2a yü 1 zü 2 1a dik 1 2a ku 1 zey 2 2a yü 1 zü 2 4a me 1 yıl 2 l 3 d r 4 .

4.2. Syllable Sequential Labeling Results

We looked at the state-of-the-art model's outputs in three stages. Specifically, the comparative analysis focused on the success rate of ambiguous words, which provided insight into this investigation.

4.2.1. Character-Based Result

If a word in a sentence is not an ambiguous word, the transformation for that word pattern will always be from one input set of syllables to only one output set of syllables. This mathematically defines a function from one set of words to another set of words. This is a simple mapping process and can be easily learned by artificial intelligence.

In fact, it can be easily performed with the help of a morphological analyzer without the need for an artificial intelligence application.

The rate of ambiguous words in the corpus used for training, validation and testing is approximately 9.72 percent. In the experiment performed on a specific test set with this ratio for ambiguous words, it was calculated what percentage of the characters were correctly predicted by the transformer. The correct prediction success rate is 99.02%. The probabilities of the prediction being correct for a particular character are presented in Table 4.2.

Table 4.2: Comparing the study of character-based results via the study of Ekici H. (2021).

Prediction	Ekici H. (2021) Accuracy of Prediction	Accuracy of Prediction in This Study
i	89.41 %	99.08 %
u	94.38 %	99.15 %
o	94.61 %	99.13 %
s	95.98 %	99.25 %
c	93.35 %	98.87 %
g	97.59 %	99.95 %
ı	97.63 %	99.39 %
ü	95.44 %	97.61 %
ö	96.39 %	97.31 %
ş	88.18 %	98.60 %
ç	97.53 %	99.05 %
ğ	88.06 %	99.29 %
Overall Accuracy	94.51 %	99.02 %

It was also noted that no modifications were found for the other letters in the Turkish alphabet, with the exception of diacritical characters. Consequently, the baseline result, which has the same accuracy rate as the diacritical character result (99.02), can be referred to as the general character-based result, which covers the complete alphabet.

4.2.2. Word-Based Result

There are a total of 28080 words containing critical letters in the studied test dataset. 27632 words were correctly predicted by the transformer. The correct prediction rate is 98.40%. This result will contribute to establishing a baseline for future studies.

4.2.3. Ambiguous Words-Based Result

If more than one meaningful word occurs due to different diacritical characters within a word, these examples are called ambiguous words. If a meaningful word is provided for only one variation of diacritic characters, the ideal word is not ambiguous, and the diacritic characters can be determined based on this variation. Using a simple algorithm, correct diacritic characters can be determined for most words without employing any machine learning algorithms. First, for a given word input, all possible words formed with every diacritical character variation are identified. If only one of these words is meaningful, that word is selected. The diacritic letters in the selected word are considered the correct choices. In experiments using a corpus, it was observed that 90.12% of diacritic characters in a text were not part of the ambiguous word class. This means that even with the above algorithm alone, a 90.12% diacritic correction success rate can be achieved without using any machine learning algorithms. Therefore, the primary measure of success for this study will be the success of diacritic correction for the 9.87% of ambiguous words.

Many applications can check whether a word is a valid word for a language. We used a morfologic analayzer tool called “TRmorph”¹ for this purpose [Cöltekin, 2010]. The morphological analyzers are generally rule-based systems and implemented using finite state transducers (FSTs). Using this tool, ambiguous words in the corpus were found and the success of the transformer on these words was investigated.

A test scenario was prepared just to test *the* success of ambiguous words. First, sentence examples containing ambiguous words were eliminated using the “TRmorph” morphological analyzer. This test set of sentences contains 3437 ambiguous words. These sentences were then presented as syllable-based input to the trained transformer. Then, with post-processing, the syllable-based output data was converted into words. The accuracy rate was investigated on the transformer response for only ambiguous words. It was observed that 97.06 % of the ambiguous words were

formed in the correct form. This rate is an indicator of high success. It also shows the success of the system in semantic learning.

A study compared the success of certain methods in terms of the accuracy of correcting ambiguous words. In the study, both statistical and machine learning-based methods were compared for 375 ambiguous words in the test set. The method proposed in this study achieved greater success than all these methods in the evaluation of 3437 words.

Table 4.3: Comparing the study for ambiguous words results via the study of Akgül et al. (2013).

Data	No	Method	Accuracy Rate
For 375 ambiguous words in the cited study.	1	AdaBoost - Local	64,43%
	2	AdaBoost - Global	91,96%
	3	J48 - Local	76,86%
	4	J48 - Global	83,20%
	5	Naive Bayes - Local	91,69%
	6	Naive Bayes - Global	94,01%
	7	1-Gram	68,03%
	8	2-Gram	76,19%
	9	3-Gram	76,85%
For the 3437 ambiguous words in this study.	10	The Proposed Method In This Study	97,06%

¹ <https://github.com/coltekin/TRmorph>

5. CONCLUSIONS

This study introduces a solution for addressing the Diacritic Restoration (DR) problem commonly encountered in digital texts, based on a sequence-to-sequence transformer model. We have presented a novel approach grounded in syllable-based analysis, and our system has undergone extensive experimentation. Our study leveraged a large corpus sourced from Wikipedia and employed a morphological analyzer to eliminate out-of-vocabulary (OOV) words. Since data cleaning is done in many different steps, no noise-related problems occurred.

In addition, most studies in Turkish have been either character-based or word-based. The morphological structure of the Turkish language has been taken into consideration, and thus, our study stands out one step further by using the syllable study model on the Turkish language for the first time. A state-of-the-art model was obtained with the special labeling method in Experiment 4. Consequently, our results surpass the achievements of existing literature, especially in comparison to n-gram and character-based methodologies.

Furthermore, our research scrutinized character-level performance, distinguishing between characters prone to errors and those displaying exceptional accuracy. Our approach is distinguished by its content-awareness, taking into consideration the semantic structure while bypassing traditional n-gram techniques.

While our investigations primarily concentrated on the Turkish language, it is worth noting that this methodology can be potentially applied to other languages that adhere to syllabication, thus indicating its language-independent applicability.

Looking ahead, it can be several extensions for our future work, particularly focusing on the high incidence of ambiguous words in the corpus. This offers an opportunity for fine-tuning our system and enhancing its performance further.

Moreover, standardizing and automating the labeling procedure is another primary goal. It is hoped that by doing this, the tagging structure can be used more extensively and provide light on future research projects that need to be completed in Turkish.

In order to train the transformer model for a future study, we first trained the diacritical characters in our current study so that they would always be the exact counterparts of the diacritical characters that should be in Turkish. Alternatively, determining what

the transformer will predict if we randomly teach the correct forms—sometimes correctly and sometimes incorrectly—could be a significant area of research. This will further unveil the transformer's capacity for learning and prediction.

This research can also be used for other NLP tasks that require labeling and spelling. The outcomes for languages acceptable for spelling and Turkish might be better than the ones that are shown right now

.

REFERENCES

- Okur, B. Ç., Takçi, H., & Akgül, Y. S. (2013, April). Rewriting Turkish texts written in English alphabet using Turkish alphabet. In *2013 21st Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE.
- Yıldırım, S., & Yıldız, T. (2015). An unsupervised text normalization architecture for Turkish language. *Research in Computing Science*, 90, 183-194.
- Özge, A. T., Bozal, Ö., & Özge, U. (2022). Diacritics correction in Turkish with context-aware sequence to sequence modeling. *Turkish Journal of Electrical Engineering and Computer Sciences*, 30(6), 2433-2445.
- Ozer, Z., Ozer, I., & Findik, O. (2018). Diacritic restoration of Turkish tweets with word2vec. *Engineering Science and Technology, an International Journal*, 21(6), 1120-1127.
- Alkhatlan, A., Kateb, F., & Kalita, J. (2020, March). Attention-based sequence learning model for Arabic diacritic restoration. In *2020 6th Conference on Data Science and Machine Learning Applications (CDMA)* (pp. 7-12). IEEE.
- Masala, M., Ruseti, S., & Dascalu, M. (2020, December). Robert—a romanian bert model. In *Proceedings of the 28th International Conference on Computational Linguistics* (pp. 6626-6637).
- Orife, I. (2018). Attentive Sequence-to-Sequence Learning for Diacritic Restoration of Yor\ub\`a Language Text. *arXiv preprint arXiv:1804.00832*.
- Nga, C. H., Thinh, N. K., Chang, P. C., & Wang, J. C. (2019, December). Deep learning based Vietnamese diacritics restoration. In *2019 IEEE international symposium on multimedia (ISM)* (pp. 331-3313). IEEE.
- Nuțu, M., Lőrincz, B., & Stan, A. (2019, September). Deep learning for automatic diacritics restoration in Romanian. In *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)* (pp. 235-240). IEEE.
- Hung, B. T. (2018, November). Vietnamese diacritics restoration using deep learning approach. In *2018 10th International Conference on Knowledge and Systems Engineering (KSE)* (pp. 347-351). IEEE.

Nguyen, K. H., & Ock, C. Y. (2010, August). Diacritics restoration in vietnamese: letter based vs. syllable based model. In *Pacific Rim International Conference on Artificial Intelligence* (pp. 631-636). Berlin, Heidelberg: Springer Berlin Heidelberg.

Alqahtani, S., Mishra, A., & Diab, M. (2020). A multitask learning approach for diacritic restoration. *arXiv preprint arXiv:2006.04016*.

Tran, Q. L., Lam, G. H., Duong, V. B., & Do, T. H. (2021, July). A Study on Diacritic Restoration Problem in Vietnamese Text using Deep Learning based Models. In *2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)* (pp. 306-310). IEEE.

Náplava, J., Straka, M., & Straková, J. (2021). Diacritics Restoration using BERT with Analysis on Czech language. *arXiv preprint arXiv:2105.11408*.

Dang, T. D. A., & Nguyen, T. T. T. (2020, October). TDP–A hybrid diacritic restoration with transformer decoder. In *Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation* (pp. 76-83).

Cöltekin, C. (2010, May). A Freely Available Morphological Analyzer for Turkish. In *LREC* (Vol. 2, pp. 19-28).

Adali, K., & Eryiğit, G. (2014, April). Vowel and diacritic restoration for social media texts. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)* (pp. 53-61).

Stankevičius, L., Lukoševičius, M., Kapočūtė-Dzikienė, J., Briedienė, M., & Krilavičius, T. (2022). Correcting diacritics and typos with a ByT5 transformer model. *Applied Sciences*, 12(5), 2636.

Náplava, J., Straka, M., Straňák, P., & Hajic, J. (2018, May). Diacritics restoration using neural networks. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*.

Ekici H., (2021), “Diacritic Restoration of Turkish Sentences”, Master Thesis, Galatasaray University.

Suppa M., (2018), “Diacritic Restoration for Slovak Texts Using Deep Neural Networks”, Master Thesis, Comenius University.

Web 1, (2024), https://en.wikipedia.org/wiki/Word_n-gram_language_model, (Erişim Tarihi: 19/06/2024).

Web 2, (2024), <https://en.wikipedia.org/wiki/Diacritic>, (Erişim Tarihi: 19/06/2024).

Yarowsky, D. (1999). A comparison of corpus-based techniques for restoring accents in Spanish and French text. In *Natural language processing using very large corpora* (pp. 99-120). Dordrecht: Springer Netherlands.

Jurafsky, D., & Martin, J. H., (2024), “Speech and Language Processing”, Online: <https://web.stanford.edu/~jurafsky/slp3>.

Almutiri, T., & Nadeem, F. (2022). Markov models applications in natural language processing: a survey. *Int. J. Inf. Technol. Comput. Sci*, 2, 1-16.

Myers, D. S., Wallin, L., & Wikström, P. (2017). An introduction to Markov chains and their applications within finance. *MVE220 Financial Risk: Reading Project*, 26.

Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics*, 37(6), 1554-1563.

Mosteller, F., Wallace, D. L., & Nerbonne, J. A. (1964). Inference and disputed authorship: The Federalist. (*No Title*).

Bayes, T. (1763). LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical transactions of the Royal Society of London*, (53), 370-418.

Cocks, J. (2011, December). A word-based approach for diacritic restoration in Māori. In *Proceedings of the Australasian Language Technology Association Workshop 2011* (pp. 126-130).

Kazama, J. I., & Tsujii, J. I. (2005). Maximum entropy models with inequality constraints: A case study on text categorization. *Machine Learning*, 60, 159-194.

Lafferty, J., McCallum, A., & Pereira, F. (2001, June). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Icml* (Vol. 1, No. 2, p. 3).

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4), 77-84.

Negara, E. S., Triadi, D., & Andryani, R. (2019, October). Topic modelling twitter data with latent dirichlet allocation method. In *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)* (pp. 386-390). IEEE.

Xiao, J., & Zhou, Z. (2020, June). Research progress of RNN language model. In *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)* (pp. 1285-1288). IEEE.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010, September). Recurrent neural network based language model. In *Interspeech* (Vol. 2, No. 3, pp. 1045-1048).

Hamad, R. A., Kimura, M., Yang, L., Woo, W. L., & Wei, B. (2021). Dilated causal convolution with multi-head self attention for sensor human activity recognition. *Neural Computing and Applications*, 33, 13705-13722.

Yao, L., & Guan, Y. (2018, December). An improved LSTM structure for natural language processing. In *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)* (pp. 565-569). IEEE.

Al-Qerem, A., Raja, M., Taqatqa, S., & Sara, M. R. A. (2024). Utilizing Deep Learning Models (RNN, LSTM, CNN-LSTM, and Bi-LSTM) for Arabic Text Classification. In *Artificial Intelligence-Augmented Digital Twins: Transforming Industrial Operations for Innovation and Sustainability* (pp. 287-301). Cham: Springer Nature Switzerland.

Santur, Y. (2019, September). Sentiment analysis based on gated recurrent unit. In *2019 International Artificial Intelligence and Data Processing Symposium (IDAP)* (pp. 1-5). IEEE.

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of big Data*, 8, 1-74.

Afridi, T. H., Alam, A., Khan, M. N., Khan, J., & Lee, Y. K. (2021). A multimodal memes classification: A survey and open research issues. In *Innovations in Smart Cities Applications Volume 4: The Proceedings of the 5th International Conference on Smart City Applications* (pp. 1451-1466). Springer International Publishing.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Tixier, A. J. P. (2018). Notes on deep learning for nlp. *arXiv preprint arXiv:1808.09772*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Montesinos, D. M. (2020). Modern methods for text generation. *arXiv preprint arXiv:2009.04968*.

Rahali, A., & Akhloufi, M. A. (2023). End-to-end transformer-based models in textual-based NLP. *AI*, 4(1), 54-110.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding with unsupervised learning.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140), 1-67.

Vickers, P., Barrault, L., Monti, E., & Aletras, N. (2024). We Need to Talk About Classification Evaluation Metrics in NLP. *arXiv preprint arXiv:2401.03831*.

Ungurean, C., Burileanu, D., Popescu, V., Negrescu, C., & Dervis, A. (2008). Automatic diacritic restoration for a TTS-based e-mail reader application. *UPB Scientific Bulletin, Series C*, 70(4), 3-12.

Zou, C., & Hou, D. (2014, September). LDA analyzer: A tool for exploring topic models. In *2014 IEEE International Conference on Software Maintenance and Evolution* (pp. 593-596). IEEE.

Azeraf, E., Monfrini, E., & Pieczynski, W. (2023). Linear chain conditional random fields, hidden Markov models, and related classifiers. *arXiv preprint arXiv:2301.01293*.

Chiu, J. T., & Rush, A. M. (2020). Scaling hidden Markov language models. *arXiv preprint arXiv:2011.04640*.

Campbell, J. C., Hindle, A., & Stroulia, E. (2015). Latent Dirichlet allocation: extracting topics from software engineering data. In *The art and science of analyzing software data* (pp. 139-159). Morgan Kaufmann.

Baimakhanbetov, M. (2023, May). Determination of the Optimal Number of Topics in the LDA Model When Working with Large Arrays of Text Data. In *2023 IEEE International Conference on Smart Information Systems and Technologies (SIST)* (pp. 332-336). IEEE.

Yao, Y., & Huang, Z. (2016). Bi-directional LSTM recurrent neural network for Chinese word segmentation. In *Neural Information Processing: 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part IV* 23 (pp. 345-353). Springer International Publishing.

Su, Y., & Kuo, C. C. J. (2019). On extended long short-term memory and dependent bidirectional recurrent neural network. *Neurocomputing*, 356, 151-161.

Gillioz, A., Casas, J., Mugellini, E., & Abou Khaled, O. (2020, September). Overview of the Transformer-based Models for NLP Tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)* (pp. 179-183). IEEE.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140), 1-67.

Lopes, A., Nogueira, R., Lotufo, R., & Pedrini, H. (2020). Lite training strategies for Portuguese-English and English-Portuguese translation. *arXiv preprint arXiv:2008.08769*.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., ... & Raffel, C. (2020). mT5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Mastropaolo, A., Scalabrino, S., Cooper, N., Palacio, D. N., Poshyvanyk, D., Oliveto, R., & Bavota, G. (2021, May). Studying the usage of text-to-text transfer transformer to support code-related tasks. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (pp. 336-347). IEEE.

Delobelle, P., Winters, T., & Berendt, B. (2020). Robbert: a dutch roberta-based language model. *arXiv preprint arXiv:2001.06286*.

BIOGRAPHY

Öznur ACAR graduated from Erzincan Nevzat Ayaz Science High School in 2006, she graduated from university with a Bachelor's degree in Computer Engineering at Izmir Institute of Technology in 2012. In 2012, she attended the Erasmus Exchange Program at Technische Hochschule Mittelhessen University of Applied Sciences in Giessen/Germany. After graduation, she started her professional career as a Software Engineer for the Turkish Economy Bank IT for 6 years. Her second job experience was working for Turkcell Technology as a Senior Software Engineer for 4 years. While working on Turkcell Technology, she started a Master's program at Gebze Technical University in 2020. She has been working for Bosch Netherlands as a Senior Software Engineer since 2023.

PUBLICATIONS AND PRESENTATIONS FROM THE THESIS

Acar Ö., Coşkun Y., Akgül Y., (2024), “Syllable-Based Context-Aware Sequence to Sequence Transformer Model for Turkish Diacritic Restoration”, The 32nd IEEE Conference on Signal Processing and Communications Applications (SIU), Mersin, Turkey, 15-18 May.