İZMİR BAKIRÇAY ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ A.B.D.

PERFORMANCE COMPARISON OF DEEP LEARNING MODELS IN TURKISH TEXT GENERATION

YÜKSEK LİSANS TEZİ

Murat Güzel

Tez Danışmanı: Prof. Dr. Adil Alpkoçak

Haziran 2023

PERFORMANCE COMPARISON OF DEEP LEARNING MODELS IN TURKISH TEXT GENERATION

Yüksek Lisans Tezi

Murat Güzel İzmir 2023

PERFORMANCE COMPARISON OF DEEP LEARNING MODELS IN TURKISH TEXT GENERATION

Murat Güzel

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Prof.Dr. Adil Alpkoçak

İzmir
İzmir Bakırçay Üniversitesi
Lisansüstü Eğitim Enstitüsü
Haziran 2023

JÜRİ VE ENSTİTÜ ONAYI

İzmir Bakırçay Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim dalında öğrenim görmekte olan Murat Güzel' in "Türkçe Metin Üretiminde Derin Öğrenme Modellerinin Performansının Karşılaştırılması" başlıklı tezi …/…/20.. tarihinde aşağıdaki jüri tarafından değerlendirilerek "İzmir Bakırçay Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği"nin ilgili maddeleri uyarınca, Anabilim dalında Yüksek Lisans olarak kabul edilmiştir.

Jüri Üyeleri	**Unvanı Adı Soyadı**	**İmza**
Üye (Tez Danışmanı)	Prof. Dr. Adil Alpkoçak	
Üye	Dr. Emre Şatır	
Üye	Dr. Nihan Özbaltan	
Üye		
Üye		

Prof. Dr. Serkan ÇINARLI
Lisansüstü Eğitim Enstitüsü Müdürü

FINAL APPROVAL FOR THESIS

This thesis titled "Performance Comparison of Deep Learning Models in Turkish Text Generation" has been prepared and submitted by Murat Güzel in partial fulfilment of the requirements in "İzmir Bakırçay University Directive on Graduate Education and Examination" for the Degree of Master of Science in Computer Engineering Department has been examined and approved on …./…./2023

Committee Members	Title, Name and Surname	Signature
Member (Supervisor)	Prof. Dr. Adil ALPKOÇAK	
Member	Dr. Emre Şatır	
Member	Dr. Nihan Özbaltan	
Member		
Member		

Prof. Dr. Serkan ÇINARLI
Director of Graduate Education Institute

ÖZET

TÜRKÇE METİN ÜRETMİNDE DERİN ÖĞRENME MODELLERİNİN PERFORMANSININ KARŞILAŞTIRILMASI

Murat Güzel

Bilgisayar Mühendisliği Anabilim Dalı

İzmir Bakırçay Üniversitesi, Lisansüstü Eğitim Enstitüsü, Haziran 2023

Danışman: Prof. Dr. Adil Alpkoçak

Metin Üretimi, otomatik olarak insan yazısına benzeyen yeni metin oluşturmak için genellikle yapay zeka ve sinir ağları, transformers ve üretken derin öğrenme modelleri (GAN) gibi teknikler tarafından desteklenen algoritmaları kullanan Doğal Dil İşleme'de (NLP) bir çalışma alanıdır. Bu çalışmada, Uzun Kısa Süreli Bellek (LSTM), Transformer tabanlı modeller ve dil modellerindeki son gelişmeler ile GPT dahil olmak üzere çeşitli son teknoloji derin öğrenme modellerinin performansını değerlendiriyoruz. Kapsamlı değerlendirmemiz, oluşturulan metnin akıcılığını, tutarlılığını ve genel kalitesini belirlemek için perplexity, BLEU puanı gibi birden çok temel metriği kapsar. Kapsamlı bir Türkçe metin veri setlerini düzenleyerek ve ön işleme tabi tutarak, değerlendirmenin Türk dilinin karmaşıklığını ve özelliklerini doğru bir şekilde yansıtan veriler üzerinde yapılmasını sağlayarak benzersiz bir katkı sağlıyoruz. Sonuçlarımız, her bir derin öğrenme modelinin göreli güçlü ve zayıf yönlerini aydınlatarak, bunların Türkçe metin üretimine uygulanabilirliğine dair içgörüler sağlıyor. Bu çalışma, yalnızca titiz bir karşılaştırmalı çalışma sunmakla kalmıyor, aynı zamanda Türkçe metin üretimi için derin öğrenme modellerinin performansını artırmaya yönelik daha fazla araştırma yapılmasının önünü açmayı umuyor. Çalışmamız, NLP'nin sürekli gelişen çalışmalarına katkıda bulunmayı ve özellikle mevcut NLP araştırmalarında Türk dilinin temsilini ele almayı amaçlamaktadır.

Anahtar Sözcükler: Metin Üretimi, GAN, GPT, LSTM, NLP

ABSTRACT

PERFORMANCE COMPARISON OF DEEP LEARNING MODELS IN TURKISH TEXT GENERATION

Murat Güzel

Department of Computer Engineering

İzmir Bakırçay University, Graduate Education Institute, June 2023

Supervisor: Prof. Dr. Adil Alpkoçak

Text Generation is a field of study in Natural Language Processing (NLP) that uses algorithms, often powered by artificial intelligence and machine learning techniques like neural networks, transformers, and generative deep learning models, to generate new text that is automatically similar to human writing. In this study, we evaluate the performance of various state-of-the-art deep learning models, including Long Short-Term Memory (LSTM), Transformer-based models, and the recent development in language models, GPT. Our comprehensive evaluation spans across multiple key metrics such as perplexity, BLEU score to determine the fluency, coherence, and overall quality of generated text. We provide a unique contribution by curating and preprocessing a substantial Turkish text dataset, ensuring that the evaluation is conducted on data that accurately reflects the complexity and characteristics of the Turkish language. Our results illuminate the relative strengths and weaknesses of each deep learning model, providing insights into their applicability to Turkish text generation. This work not only offers a rigorous comparative study but also hopes to pave the way for further research in enhancing the performance of deep learning models for Turkish text generation. Our study is aimed at contributing to the ever-evolving landscape of NLP, and particularly, to address the under-representation of the Turkish language in current NLP research.

Keywords: Text Generation, GAN, GPT, LSTM, NLP

.../.../20...

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın İzmir Bakırçay Üniversitesi tarafından kullanılan ……. bilimsel intihal tespit programıyla tarandığını ve hiçbir şekilde "intihal içermediğini" beyan ederim. Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

Murat Güzel

.

.../.../20...

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with …… scientific plagiarism detection program used by İzmir Bakırçay University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

Murat Güzel

Contents

Acronyms

ADAM Adaptive Moment Estimation

AI Artificial Intelligence

ANN Artificial Neural Network

BERT Bidirectional Encoder Representations from Transformers

BLEU Bilingual Evaluation Understudy

CNN Convolutional Neural Network

DL Deep Learning

GAN Generative Adversarial Network

GPT Generative Pre-trained Transformer

GRU Gated Recurrent Unit

GSGAN GAN with the Gumbel-softmax Distribution

LSTM Long Short-Term Memory

ML Machine Learning

NLP Natural Language Processing

NMT Neural Machine Translation

PPL Perplexity

RNN Recurrent Neural Network

Seq2Seq Sequence-to-Sequence

VAE Variational Autoencoder

List of Figures

List of Tables

1. INTRODUCTION

Text generation is a field of study in Natural Language Processing (NLP) that uses algorithms, often powered by artificial intelligence and machine learning techniques such as neural networks, transformers and generative deep learning models to generate new text that is similar to human writing. These models use large amounts of data and complex algorithms to create text. They have a wide range of uses, including chatbots, virtual assistants, and automated content creation (Fatima et al., 2022).

The primary objective of text generation systems is to generate text that is contextually precise, meaningful, and coherent based on a given prompt that has a set of constraints or keywords. With this process, these systems can assist in various tasks, including but not limited to drafting reports, generating online messages, and even producing creative writing such as poetry and narratives. In addition, it is possible to train text generation models on particular domains, such as medical reports or legal documents. In general, the potential of text generation technology to transform our language interactions is noteworthy, and it is progressively gaining more importance in our daily activities. (Celikyilmaz et al., 2020).

The advancement of deep learning methods and state-of-the-art language models like Generative Pre-trained Transformer (GPT) and its successors has led to the creation of text generation systems that can better generate coherent and contextually relevant material (OpenAI, 2023). Text generation tools may boost productivity by helping people express themselves more accurately, generate new ideas, and produce written material more quickly. Marketing, journalism, entertainment, healthcare, education, and customer service are just a few of the many fields that may benefit from text generation. It is helpful for a wide variety of tasks, including but not limited to content personalization, translation, question answering, agent creation, and chatbot development. Since there are still many issues to be addressed and opportunities for innovation, it is an exciting and dynamic area to research (Iqbal and Qureshi, 2022).

The field has been revolutionized in recent years thanks to deep learning techniques and generative models (Montesinos, 2020). The emergence of text generation through deep learning can be attributed to the development of Recurrent Neural Network (RNN)

(Elman, 1990), specifically the introduction of Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997). These architectures played a crucial role in modeling long-range dependencies in sequences, which was essential for addressing the complexity of natural language. Further advancements in deep learning enhanced text generation capabilities, such as introducing the attention mechanism and the Transformer architecture (Vaswani et al., 2017) (Bahdanau et al., 2014). These innovations allowed the discovery of complex patterns and relationships within extensive language datasets, ultimately leading to more coherent and context-sensitive text generation. A crucial element of text generation has been the development of generative models, including Variational Autoencoder (VAE) (Kingma and Welling, 2022), Generative Adversarial Network (GAN) (Goodfellow et al., 2014), and more recently, autoregressive models like GPT (Radford and Narasimhan, 2018). These models have shown remarkable results in producing text that resembles human writing while tackling issues such as mode collapse, high dimensionality, and model overfitting (Iqbal and Qureshi, 2022).

1.1. Motivation

Although considerable research has been devoted to text generation rather less attention has been paid to specifically Turkish text generation. In this thesis, our focus is on comparing the performance of deep learning models in Turkish text generation. Turkish, with its agglutinative and morphologically rich structure, presents unique challenges for NLP tasks. The complex morphology and distinctive syntactic properties of Turkish have made accurate modeling a challenging task for traditional NLP techniques. However, recent advances in deep learning have provided new opportunities to address this challenge.

The Turkish language has distinct linguistic features, such as agglutination and vowel harmony, which can pose challenges for deep learning models. Investigating how these models handle the complexities of the Turkish language will provide insights into the factors that influence their performance and suggest potential areas for improvement.

This thesis aims to provide valuable insights into the strengths and weaknesses of each model, thereby facilitating more effective NLP applications for the Turkish language. The findings can inspire new approaches and techniques that advance the state-of-the-art in Turkish NLP, driving innovation and progress in the field. Additionally, this thesis

aims to provide insights that will assist linguists, researchers, and application developers in achieving more effective and successful results in the field of Turkish text production and natural language processing.

1.2. Research Questions

In the research question section of the thesis, we want to provide a complete picture of the main questions that will guide our study on how well deep learning models perform in the context of Turkish text generation. Research questions include not only the specific models and architectures used but also different factors that may affect how well they perform and Turkish language features that may impact the results. By answering these research questions, we hope to learn more about the strengths and weaknesses of deep learning models in generating Turkish text and to identify areas where further research and development can lead to better results and performance.

- How do different deep learning architectures (LSTM, Encoder Decoder, GPT, GAN) perform when generating Turkish text, and what are their strengths and weaknesses?
- How do factors such as training data size, preprocessing techniques, and model complexity impact the performance of deep learning models for Turkish text generation?
- How do state-of-the-art language models compare to traditional deep-learning models like RNN methods for Turkish text generation, and how does their performance vary across different text domains (e.g. Bible, Recep Tayyip Erdogan speeches, Mesnevi - Mevlana)?
- What specific linguistic characteristics of the Turkish language affect deep learning models' performance in text generation tasks, and do character-based or word-based approaches yield more accurate results?

1.3. Thesis Organization

The thesis will start by describing definitions and terms in deep learning and evaluation metrics. The next chapter starts by comprehensive literature review of the current state-of-the-art deep learning models along with their applications in text generation and NLP, such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, Generative Adversarial Networks (GAN) and the Transformer architecture like GPT. The methodology section will also compare and analyze the performance of the selected deep learning models for Turkish text generation. We will identify the strengths and weaknesses of each model and investigate the impact of different model architectures with carachter-based text generation and word-based text generation, hyperparameters, and training strategies on text generation quality. Comparing performance outlines the evaluation metrics employed in this thesis, including Perplexity (PPL) and Bilingual Evaluation Understudy (BLEU), to provide a comprehensive understanding of the models' effectiveness.

In the last chapter, we summarize the findings and provide insights into the implications of this thesis for future work in Turkish text generation.

2. DEFINITIONS AND TERMS

In this chapter of the thesis, we aim to clarify and establish the fundamental concepts and terminology that are crucial for comprehending the different facets of deep learning models within the context of Turkish text generation. We provide definitions and explanations for the key features of each deep learning model whose performance we compared, highlighting their respective advantages and disadvantages. Additionally, we analyze the essential activation functions and learning from data methods employed during the training phase for each model. Furthermore, we define and explain the performance measurement methods employed to evaluate the models' performance in this thesis.

2.1. Deep Learning

Deep Learning (DL) is a subfield of Artificial Intelligence (AI) and Machine Learning (ML) that focuses on the development and utilization of artificial neural networks to process, analyze and learn from data. These neural networks, inspired by the structure and function of the human brain, are capable of automatically learning complex patterns and representations from raw input data, such as images, audio, or text. With the ability to identify and process patterns, these networks are able to tackle complicated tasks and make decisions with minimal human involvement. The deep learning process consists of forming multilayered neural networks that have the ability to learn from data (Lecun et al., 2015).

The foundation of deep learning is the architecture of a feedforward neural network called multi-layer perceptron. The perceptron is an elementary model of an artificial neuron, initially proposed by Frank Rosenblatt in 1958, that serves as a binary classifier capable of learning linearly separable patterns from input data. The perceptron algorithm processes the input data by computing a weighted sum of the input features, incorporating a bias term, and subsequently applying a binary threshold function to generate an output signal corresponding to one of two distinct classes (Rosenblatt, 1958).

Deep learning is developed from a neural perspective and mimics biological neurons. Inputs are evaluated in specific relationships with the output. With these models, n

input parameters such as $x_1, x_2, ..., x_n$ are associated with their output value y. This association is made by multiplying them with weights such as $w_1, w_2, ..., w_n$ and adding a certain bias value (Goodfellow et al., 2016).

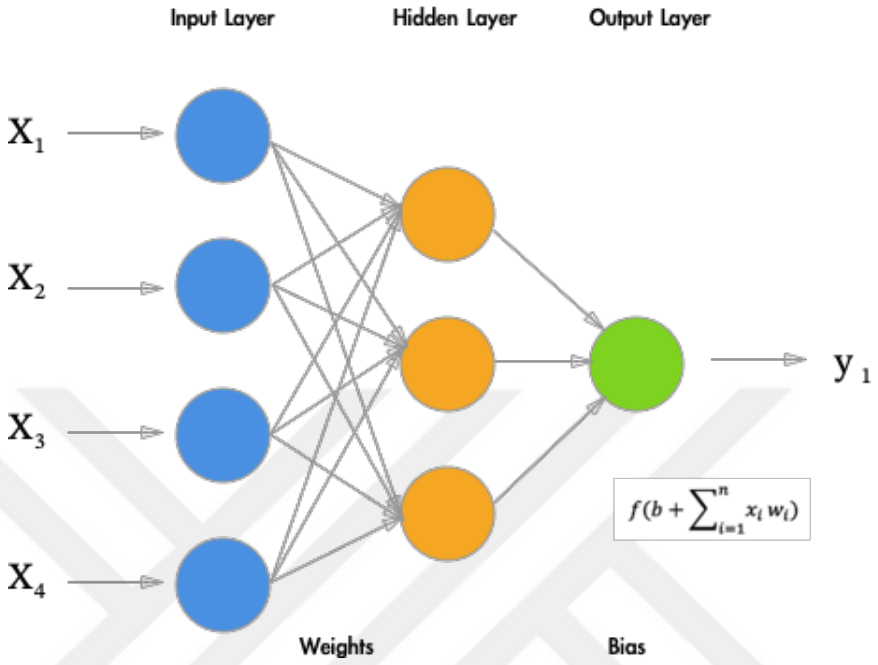

Figure 2.1: Feed Forward Neural Network

Artificial Neural Network (ANN) consist of interconnected nodes, arranged in a graph-like structure, which are biologically inspired by brain neurons. These nodes are organized into layers connected in sequence. An output layer is the final layer in a series of layers of a feedforward neural network. Following the input layer, subsequent layers are referred to as hidden layers. An activation function, a mathematical function employed in artificial neural networks, introduces non-linearity to the output of an artificial neuron or layer. The primary purpose of activation functions is to transform the input data into the output data while determining the activation level. The output of the activation functions serves as input parameters for the subsequent layer, allowing learning of non-linear relationships (Zurada, 1992).

To effectively learn the relationship between input and output signals in activation functions, it is crucial to identify and utilize the most accurate weights between layers. The output parameter, computed by the activation function using the weights employed by artificial neurons, typically does not match the actual output value. This discrepancy is known as the loss function. Minimizing the loss function is accomplished through the re-evaluation of weights, a process called backpropagation. The learning process occurs

in this manner (Lecun, 2001).

2.2. Recurrent Neural Network (RNN)

Sequence learning focuses on the modeling, and prediction of sequential data. In sequence learning tasks, the order of the elements in the input data is crucial for accurate modeling and prediction. Examples of sequence learning tasks include natural language processing, speech recognition, time series forecasting, and gesture recognition. These tasks are highly complex and require sophisticated algorithms to be solved effectively. One such task is speech recognition, where an acoustic signal must be transcribed into words or sub-word units. Recurrent neural networks (RNNs) are powerful sequence learners that would seem well suited to such tasks due to their ability to model long-term dependencies. This makes them ideal for tasks that require the prediction of sequences of labels, such as speech recognition (Graves et al., 2006). RNNs are able to generate sequence real data that are processed incrementally, with predictions made for each subsequent step. This procedure is iterated until the desired sequence length is attained. By forecasting the succeeding step according to prior inputs, the RNN can generate novel sequences that bear resemblance to the original input data (Graves, 2014).

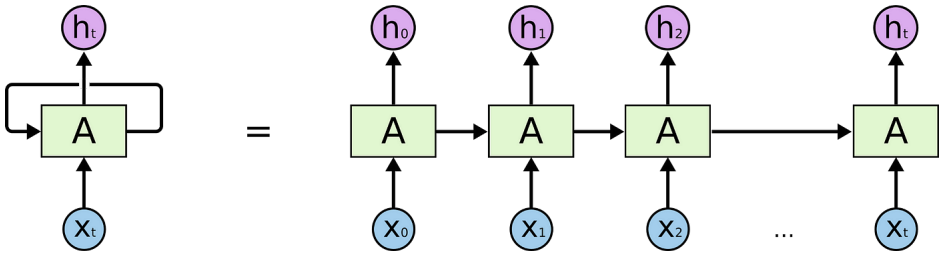

Figure 2.2: Recurrent Neural Network (RNN)

Recurrent Neural Networks operate by retaining a hidden state which is modified at each time step during the processing of the input sequence. The next time step gets the hidden state previously generated and the input at the value of the current time step.

An RNN performs the following operations at each time step (t):

- Gets the input at time step t (x_t).

- Updates the hidden state based on the current input (x_t) and the previous hidden state (h_{t-1}). This is typically done using a nonlinear activation function, such as the hyperbolic tangent (tanh) or the Rectified Linear Unit (ReLU):

  $$h_t = f(W_x * x_t + W_h * h_{t-1} + b_h)$$

  where W_x and W_h are weight matrices, and b_h is the bias vector.

- Computes the output (y_t) for the current time step

  $$y_t = g(W_y * h_t + b_y)$$

  where W_y is the output weight matrix, and b_y is the output bias vector.

- Passes the updated hidden state (h_t) to the next time step and repeats the process.

The primary obstacle when training RNNs involves addressing the vanishing and exploding gradient issue that occurs when gradients become excessively small or large during backpropagation over time. This concern is alleviated through the use of sophisticated RNN architectures like LSTM and GRU (Goodfellow et al., 2016).

2.3. Long Short-Term Memory (LSTM)

LSTM is a type of Recurrent Neural Network (RNN) architecture that was developed to overcome the disadvantages of standard RNN in long-range learning sequence data. The LSTM was first presented by Sepp Hochreiter and Jürgen Schmidhuber in 1997. Since then, it has become an essential model in many deep learning applications, especially in the fields of natural language processing, speech recognition, and time series analysis (Hochreiter and Schmidhuber, 1997).

LSTMs are composed of multiple linked LSTM cells, which are used for memory, each tasked with preserving information throughout a given duration. The regulation of these memory cells is achieved through three separate gates, namely the input gate, the forget gate, and the output gate. These gates are pivotal in dictating the movement of information within the network, enabling LSTMs to adaptively learn when to retain, adjust, or eliminate information, depending on the context of the input sequence (Gers et al., 2000).

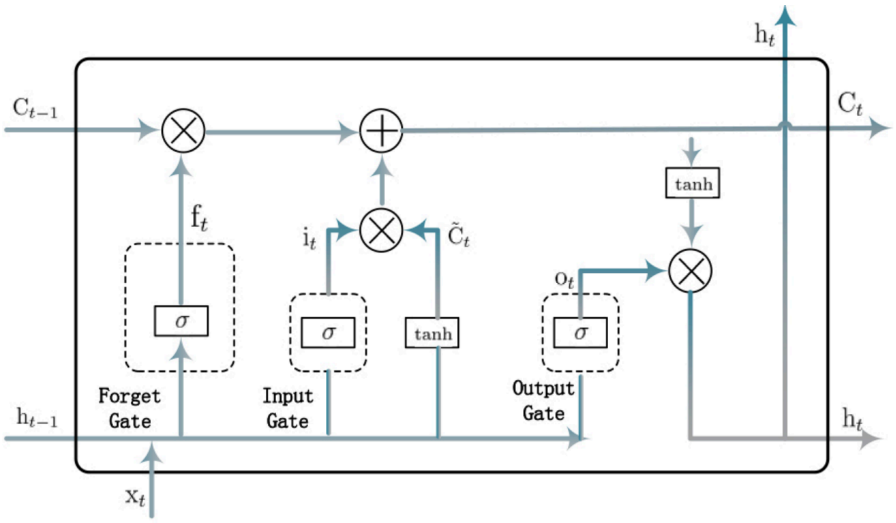

Figure 2.3: Long Short Term Memory (LSTM) (Luo et al., 2021)

Input Gate: The job of the input gate is to figure out what new information should be added to the memory cell. It is composed of a sigmoid activation function that gives out numbers between 0 and 1 to show how much new information should be kept.

Forget Gate: The role of the forget gate is to decide which pieces of information should be removed from the memory cell. Similar to the input gate, it employs a sigmoid activation function, generating values ranging from 0 to 1, which indicate the proportion of information that should be either preserved or eliminated.

Output Gate: The output gate's function is to manage the transfer of information from the memory cell to other parts of the network. By using a sigmoid activation function, it determines which segments of the memory cell's information should be conveyed as output (Goodfellow et al., 2016).

2.4. Encoder - Decoder Model

The encoder-decoder model has become a key model in Neural Machine Translation (NMT) and NLP. Its success has come from being able to make sequence-to-sequence learning models, translate between two sequence data like languages (Stahlberg, 2020).

c In the context of NMT, the encoder reads the source language, and the decoder produces the target language translation.

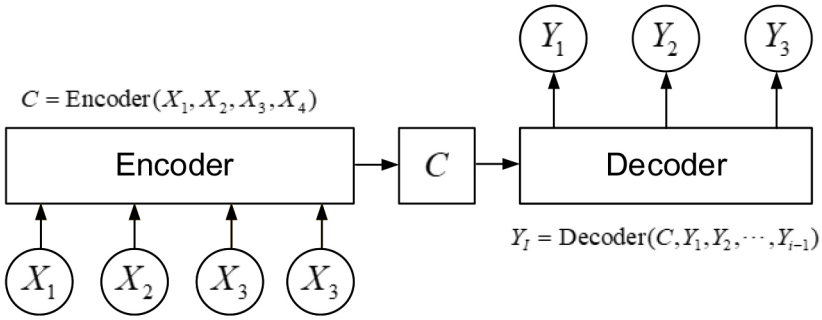

Figure 2.4: Encoder - Decoder with Context Vector (Luo et al., 2021)

The encoder, which is a neural network, processes the input sequence then produces a fixed length output which called context vector shows in figure 2.4 as *C* between encoder and decoder. The context vector plays a crucial role in capturing the semantic information of the input sequence and transferring it to the decoder to generate the output sequence. The most common type of encoder is a LSTM or Gated Recurrent Unit (GRU), which is designed to address the vanishing gradient problem in conventional RNN. When using LSTM for encoding network, the context vector consists of a combination of the LSTM hidden state and the cell state.

The docoder is an another neural network processes context vector that generated by encoder. The choice of the decoder in an encoder-decoder model depends on the specific application and desired properties of the model. Several popular types of decoders have been used with encoder-decoder models, each with its own strengths and weaknesses. Some common decoders include: RNN decoders, Transformer decoders and Convolutional Neural Network (CNN) decoders (Cho et al., 2014).

2.5. Attention Mechanism and Transformers

The attention mechanism is a widely used and highly influential technique in deep learning models, particularly in the field of natural language processing (NLP). It was first introduced by Bahdanau et al. (2014) as a method to improve the performance of Sequence-to-Sequence (Seq2Seq) models. One of the most popular applications of attention is in the context of NMT models. In these models, attention is used to align the source and target language sentences, making it possible to generate accurate translations (Niu et al., 2021).

At its core, the attention mechanism is designed to overcome the limitations of fixed-

length context vectors in traditional Seq2Seq models. In these models, the entire input sequence is encoded into a single fixed-size vector, known as the context vector, which is then used to generate the output sequence. This approach can lead to a loss of information, especially when dealing with long input sequences. The attention mechanism resolves this issue by enabling the model to weigh the importance of different input elements and focus on the most relevant parts of the input sequence during the decoding process (Vaswani et al., 2017).

On the attention mechanism, there are three main vectors called query (Q), key (K), and value (V). The query vector shows the current input sequence. It is used to figure out which of the input tokens are better for the current time step of the sequence. The query vector can be built from the input tokens or from the LSTM model's or Transformers model's hidden states. In self-attention methods, the query vectors, the key and the value vectors all come from the same input sequence. This lets the model focus on different tokens of the input sequence. The key vector represents the content of each input token and serves as a "key" to unlock the relevant parts of the input sequence. Key vectors are often generated using a trainable weight matrix that the model learns as it trains. When calculating attention scores, the model compares the query vector with each key vector to understand how important or relevant each input token is for the current context. Once the model has determined which input tokens are relevant with using the query and key vectors, it computes a weighted sum of the value vectors to create a context vector, which summarizes the relevant information from the input sequence (Brauwers and Frasincar, 2023).

The most important part of the transformer design is the attention mechanism. It lets the model give different tokens in the input sequence different weights based on how important they are in a certain situation. Transformers were first described in a paper called "Attention Is All You Need" by Vaswani et al. (2017). This paper suggested a new architecture for processing input sequences that relied only on self-attention mechanisms. The earlier sequence-to-sequence models, which used RNN or CNN, were different from this method. Even though these older models had great results, they had some problems, like not being able to handle long-range relationships well and being slow to process. The unique architecture of Transformers got around these problems, starting a revolution in the field of NLP and beyond (Vaswani et al., 2017).

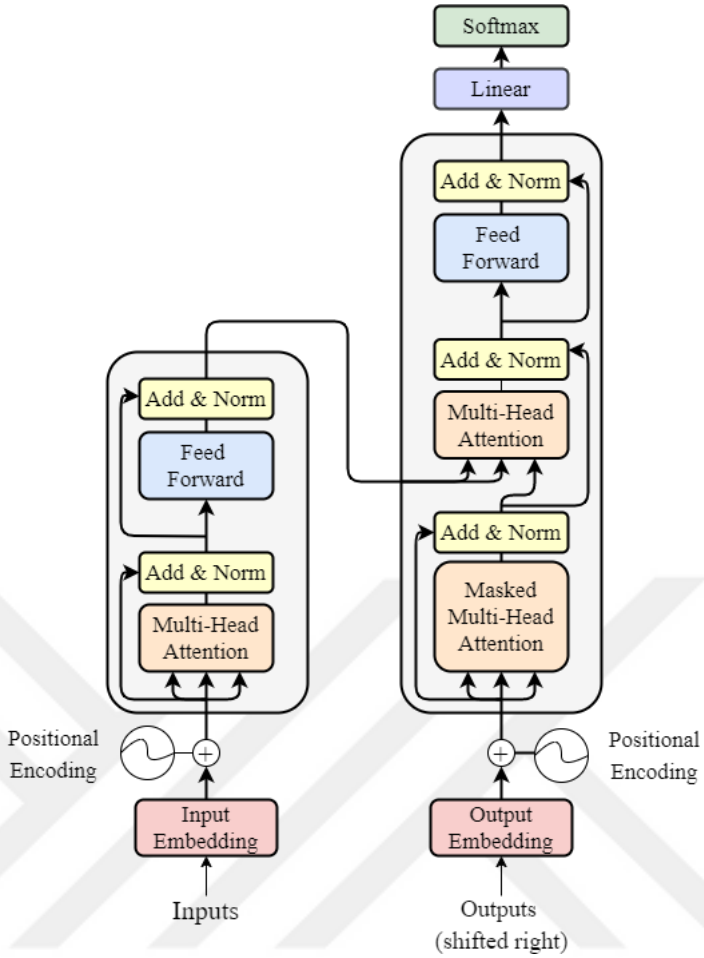

Figure 2.5: Transformers (Vaswani et al., 2017)

The encoder and the decoder are the two key parts of the transformer design, which is a popular deep learning model. Both of these parts are made up of several layers, and they work together to handle input sequences and make output sequences.

The encoder's task is to take a sequence of inputs and turn them into a continuous representation that includes important information. The encoder has several layers, and each layer has a multi-head self-attention and a position-wise feed-forward network sub-layer. The multi-head self-attention sublayer figures out attention scores for each token in the input series. This shows how the tokens are related. The output of the multi-head self-attention sublayer is given a linear change by the position-wise feed-forward network sublayer. This lets input sequences be processed in parallel. In the encoder layers, residual links and layer normalization are also used to help information flow and keep the training process stable.

The decoder's job is to turn the intermediate representation made by the encoder into an output sequence. Like the encoder, it has several layers. Each layer has a covered

multi-head self-attention sublayer, an encoder-decoder attention sublayer, and a position-wise feed-forward network sublayer. The masked multi-head self-attention sublayer is like the one in the encoder, but it also has a masking device that stops the decoder from paying attention to tokens in the output series that come after the current one. The encoder-decoder attention sublayer figures out attention scores between the output sequence that has been made so far and the intermediate representation from the encoder. This lets the processor focus on the most important parts of the input series when making the output. Lastly, the position-wise feed-forward network sublayer takes the outputs of the other sublayers and changes them in a linear way. The encoder also uses leftover links and layer normalization to help with the flow of information and make training more stable.

The output of the decoder is generated by applying a linear transformation followed by a softmax function, resulting in a probability distribution over the output vocabulary. The token with the highest probability is selected at each position in the output sequence (Vaswani et al., 2017).

2.6. Generative Adverserial Networks (GAN)

GAN, first introduced by Goodfellow et al. (2014) as a type of deep learning model designed to generate novel, never before seen synthetic data samples. Two distinct neural networks make up GAN: the generator and the discriminator. Ultimately, these networks compete in a zero-sum game to generate high-quality synthetic data.

The generator generates synthetic data samples, while the discriminator evaluates them to determine their authenticity compared to real-world data. Eventually, through continuous iteration and competition, the generator becomes more adept at producing genuine data samples, while the discriminator becomes more adept at identifying them (de Rosa and Papa, 2021).

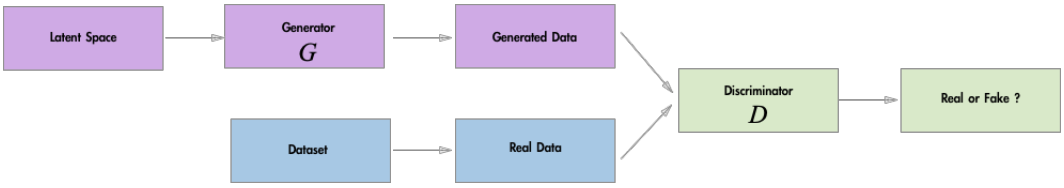

Figure 2.6: GAN Architecture

GAN function through an iterative process involving the two networks, the genera-

tor, and the discriminator.

The generator (G) creates a synthetic data sample based on random noise. G is a function that takes in a random noise vector (z), sampled from a predefined distribution generally in gaussian, and outputs a synthetic data sample.

$$G(z; \theta_g)$$

Discriminator (D) takes in data (both real data and the generated) and outputs the probability that the data is real. The discriminator's goal is to maximize this probability for real data and minimize it for generated data.

$$D(x; \theta_d)$$

The training process of GANs can be formalized as a minimax game between the generator and the discriminator. The generator aims to minimize the following objective function, while the discriminator aims to maximize it:

$$min_G max_D V(D, G) = E_{xP_{data(x)}}[logD(x)] + E_{zP_z(z)}[log(1 - D(G(z)))]$$

E denotes the expectation.
$P_{data(x)}$ represents the true data distribution.
$P_z(z)$ stands for the noise distribution from which z is sampled.
The first term, $E_{x\ P_{data}(x)}[logD(x)]$, represents the expectation of the log probability that the discriminator correctly identifies real data. The discriminator aims to maximize this term.
The second term, $E_{z\ P_z(z)}[log(1 - D(G(z)))]$, represents the expectation of the log probability that the discriminator correctly identifies generated data. The generator aims to minimize this term (Goodfellow et al., 2014).

2.7. Generative Pre-trained Transformer (GPT)

GPT is a series of artificial intelligence language models, based on the transformer architecture. GPT model has been particularly successful in natural language processing tasks. The original GPT model was created by Radford et al. (2019). Developers of GPT model have released multiple iterations of GPT, with the most recent version being GPT-4 at the time of the thesis. These models have grown increasingly more advanced and capable of understanding and generating human-like text (Radford et al., 2019).

The main idea behind the GPT model is to leverage unsupervised pre-training on large-scale corpora, followed by fine-tuning on specific tasks using supervised learning.

The GPT model uses a unidirectional context as a masked self-attention mechanism, where the attention scores are masked to prevent the model from attending to tokens that come later in the sequence. Due to this design decision, GPT can generate text autoregressively by predicting one token at a time using the context previously generated tokens have provided (Radford and Narasimhan, 2018).

The GPT model is trained on many unlabeled text data during the pre-training phase, learning to predict the next token in a sequence based on the context. Through this process, the model learns about the structure of language and its typical representation. In the fine-tuning phase, the pre-trained model is adapted to specific tasks with the help of labeled data. The architecture of this system enables it to perform effectively on various NLP tasks with minimal adjustments needed for each specific task (Radford et al., 2019).

The GPT model can be used in many different ways. For example, it can be used to generate text in the style of a particular author or to create summaries from long documents. It can also be used for translation tasks, question answering, and more. With its powerful capabilities, the GPT model is revolutionizing the field of natural language processing and making it possible to tackle a wide variety of complex tasks with less effort. Furthermore, the GPT model is becoming increasingly popular due to its ease of use and scalability. With its advances in deep learning technology, the GPT model is quickly becoming an indispensable tool for those interested in NLP.

2.8. Perplexity Quality Measurement

PPL is a commonly used metric in language models, especially in the field of text generation. Perplexity is a kind of calculation method that evaluates the quality of a language model. It calculates how well a language model can predict the next token in a time step of sequence input. A low perplexity score shows that how the language model is good at predicting the next word in the corpus. Low perplexity score means it is more likely to generate coherent and meaningful text. On the other hand high perplexity score means the language model has a low performance to predict the next word and there may be issues with the model architecture or the training data (Celikyilmaz et al., 2020).

Perplexity can be formally defined as:

$$PPL(W) = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(w_i)}$$

- PPL(W) is the perplexity calculation of a sequence of N tokens in $W = w_1 w_2 ... w_N$,
- $P(w_i)$ is the probability assigned by the language model for each token in sequences,
- The sum runs over all words in the sequence,
- And N is the total count of tokens.

Perplexity furthermore is a primarily comparative measure. It is very useful when comparing the performance of different models on the same task. In this thesis we use perplexity to compare deep learning models in text generation task (Gu et al., 2020).

2.9. BLEU Score

BLEU is a way to measure the quality of machine translations by comparing them to reference translations made by humans. The BLEU scoring method works by looking at n-grams, which are groups of n words or phrases from a sample of text or speech. The BLEU-2, BLEU-3, and BLEU-4 scores look at bigrams, trigrams, and 4-grams, respectively. This makes the language and context of the test more complicated. These scoring methods compare the machine translation to the reference translation to see how well the

sequence of two, three, or four words was kept. The Corpus-Based BLEU, on the other hand, looks at the quality of the translation as a whole group of texts. This method gives a more thorough review because it looks at the quality of the translations as a whole instead of just one sentence at a time. But it might miss some mistakes in how a sentence is translated. Even though BLEU scores can be used to measure and compare quality, it's important to remember that they should be used with other factors like fluency, cultural nuances, and general meaning retention to get a full picture of how well a translation works (Papineni et al., 2002).

3. RELATED WORKS

The field of text generation has witnessed significant advancements in recent years, with various deep learning models being proposed and applied to a wide range of tasks. Text generation methods often fall into one of two categories text-to-text and data-to-text (Gatt and Krahmer, 2018).

Text to text systems, the input and output are both text. This category includes models like Seq2Seq, GPT, Bidirectional Encoder Representations from Transformers (BERT), and other transformer based models. They are typically used for tasks such as machine translation (Wu et al., 2016), text summarization (See et al., 2017), question answering systems (Rajpurkar et al., 2018), paraphrase generation (Li et al., 2018) and chatbots (Suta et al., 2020). On the other hand, data to text systems take structured data as input and generate text as output. This may comprise tabular data, a sequence of information, a graphical representation, or any other form of non-textual data. The task assigned to the system involves parsing the structured data and producing natural language text that accurately represents the given information. These systems are commonly employed for the purpose of producing reports, descriptions, or narratives based on data (Puduppully et al., 2019). These systems, text generation with deep learning models, has evolved over the years with advancements in natural language processing and deep learning techniques.

Early days of natural language processing, rule based and statistical methods were using for text generation (Manning and Schütze, 1999). In deep learning era, RNN have been used for text generation tasks due to its ability to handle sequential data (Elman, 1990).

Since RNN's vanishing gradient problem, Peng and Yao (2015) proposed a new external memory mechanism, Chung et al. (2015) proposed a new gated layer for multiple recurrent layers, Zheng et al. (2015) used the probabilistic method as a conditional random field for RNN model. Santhanam (2020) proposed a new LSTM model with using generate context vector for context based text generation. Chakraborty et al. (2020) has shown that number of LSTM cell effects on character based text generation quality.

These models typically utilize RNNs or LSTMs to encode the input sequence and generate the output sequence. The introduction of the transformer architecture by Vaswani

et al. (2017), with titled as The Attention is All You Need, improved a significant breakthrough in text generation research. By utilizing self-attention mechanisms, transformers are capable of capturing long-range dependencies more effectively than RNNs or CNNs. This has led to a series of state-of-the-art models, including GPT-2 (Radford et al., 2019), GPT-3 (Brown et al., 2020), and GPT-4 (OpenAI, 2023) which have demonstrated remarkable performance in a wide range of NLP tasks. Another significant transformer-based model that has influenced the field of text generation is BERT. BERT's bidirectional pre-training methodology has allowed it to perform at the cutting edge in a variety of NLP tasks, and its ability to be fine-tuned has made it a popular option for many text generation applications (Devlin et al., 2019).

Another deep learning model for text generation is GAN. Although GAN is proposed for image generation, there are studies about text generations. (Goodfellow et al., 2016). Studies in GAN proposed a novel methods for converging discrete data. Kusner and Hernández-Lobato (2016) proposed a novel approach called GSGAN. In this paper, the authors addressed the problem of generating sequence of discrete elements, such as text data, using gumbel softmax for discrete data. Yu et al. (2017) introduced SeqGAN, which was the first model to extend GAN to discrete token generation. In their work, the authors focused on various domains, including Chinese poems, Nottingham Music, Obama Speech, and Synthetic Data. Lin et al. (2017) introduced RankGAN, a model that deviated from the traditional GAN setup by employing ranked data instead of real and fake data pairs. Fedus et al. (2018) introduced MaskGAN, which employed an actor-critic conditional architecture for text generation. The model was evaluated on the IMDB Movie dataset and Penn Treebank dataset. Zhang et al. (2017) proposed TextGAN, which addressed text generation by combining data from BookCorpus and ArXiv. The authors introduced a high-dimensional latent feature distribution with kernelized discrepancy.

There is a very limited number of study in Turkish language. Demir and Oktem (2023) proposed a benchmark dataset in Turkish for data to text generation models. Kutlugün and Şirin (2018) uses class based n-gram model for meaningful text generation in Turkish language.

4. METHODOLOGY

The purpose of this study is to compare the performance of various deep learning models in Turkish text generation. The methodology of this thesis involves data collection, data preprocessing, model selection, model training, evaluation and comparison.

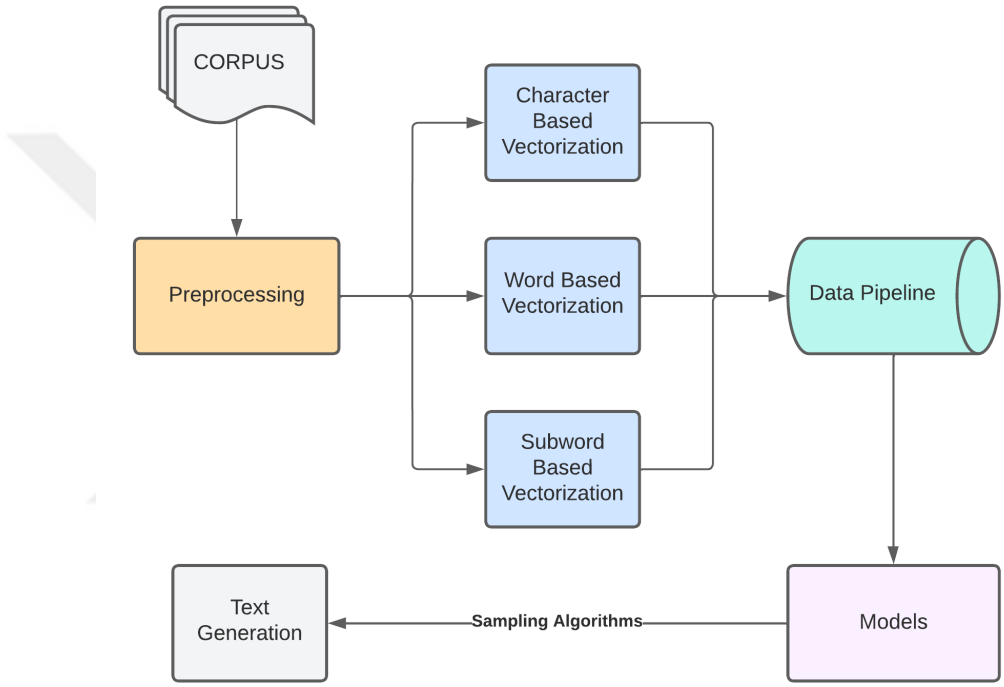

Figure 4.1: Text Generation Method

Our study as shown in Figure 4.1 leverages the sophisticated capabilities of natural language processing to generate text, a process that involves a series of interconnected steps. Firstly, we took the raw text data and subjected it to a preprocessing phase, wherein it was cleaned and transformed into a format that our algorithms could understand. This initial groundwork set the stage for all the steps that followed. Next, we tokenized the preprocessed data, breaking it into smaller units - tokens. We used various scales of tokenization, ranging from individual characters, and words to even subword units, tailoring the granularity according to our specific needs. Once tokenization was completed, we constructed a data pipeline. This organized the tokens into a structured format that could be effectively fed into our deep learning models. The fourth step involved training our cho-

sen deep learning models on this structured data. During this training phase, our model learned to predict the next token in a sequence given the preceding tokens, effectively gaining an understanding of the language's structure. The final step marked the actual generation of new text. With the aid of our trained model and a range of sampling algorithms including greedy search, beam search, top-k sampling, and temperature sampling, we were able to generate new, meaningful, and diverse text sequences, thereby successfully completing the text generation process.

4.1. Data Collection

The choice of corpora, structured sets of texts used to train models, is very important in text generation process because it affects how well the generated text works and how good it is. When selecting a corpus for text generation, there are a number of factors to consider. These factors include the size of the corpus, the type of the text and the intended use of the generated text (Xie et al., 2023).

Factors such as the size and the quality of the corpus, the type of text, and the diversity of the text are considered when choosing a corpus. A large corpus will provide more data for language models to learn, leading to more accurately and fluently composed text. The quality of the corpus was also considered. Corpora with numerous typographical errors, inconsistent formatting, or unreliable information can result in a poorly performing model. Therefore, the quality of the corpus is also an important criterion for measuring model performance. The type of text is another important factor that determines the style and content for text generation tasks. A diverse corpus will expose the text generation model to a wider range of writing styles and topics, which will make it more capable of generating different kinds of text (Radford et al., 2019). Considering these criteria, the following corpus was used in this thesis.

- Bible New Testament
- Recep Tayyip Erdoğan Speech
- Mesnevi Mevlana

These three corpora were carefully chosen to provide a balanced and comprehensive overview of the performance of the selected deep learning models. Using them, we can

assess not only the models' overall text generation capabilities, but also their ability to adapt to different styles, formats, and contextual nuances of the Turkish language.

Training and testing models on these diverse corpora allow us to gain insights into their strengths and weaknesses in different text generation tasks. This variety of sources will enhance the generalizability of our findings, ultimately leading to a more robust comparison of deep learning models' performance in Turkish text generation.

4.2. Data Preprocessing

Text preprocessing is a critical step in natural language processing and deep learning models for text generation, as it directly influences the model's performance (Camacho-Collados and Pilehvar, 2018). In our study, we adopted the following four preprocessing strategies, designed to optimize the performance of our deep learning models in Turkish text generation.

Three corpora's text data is case sensitive, the same word may be represented in different ways depending on its placement within a sentence or its usage, such as "Ankara" and "ankara". All text input was transformed into lowercase to maximize data consistency and simplify the learning process to measure models' performance (Pennington et al., 2014).

Since we want to examine how the models learn the punctuation marks, we didn't remove them. However all special characters, except punctuation marks and numbers, were removed from the text input. This resulted in a cleaner dataset, enabling the model to focus on the semantic and syntactic aspects of the text (Manning et al., 2014).

For word-based tokenization, a non-word (OOV) strategy was applied to ensure that the model could be effectively generalized to unseen or rarely used words. Words appearing below a certain frequency threshold in the training data were replaced with a specific OOV identifier for each corpus. This strategy allowed the model to learn a representation of words not encountered during training, increasing its ability to process unusual words in the test set (Camacho-Collados and Pilehvar, 2018).

Vocabulary building is an another key preprocessing step in text generation. In this study, a vocabulary was created using the preprocessed and cleaned data. In word-based tokenization, each unique word was assigned an index, facilitating the transformation of words into numerical values, a necessary step for deep learning algorithms. In character-

based tokenization each character was assined an index (Mielke et al., 2021).

Punctuation marks are not removed during language model training. The manner in which the model learns punctuation marks, such as commas and question marks, is also an important criterion for performance comparison. Table 4.1 shows the most frequently occurring words for each corpus. However, punctuation marks and stopwords have been removed for GAN models where a discriminator is used to enhance classification performance. In Table 4.2, the most frequently used words, now in their cleaned form, can be observed.

Bible - New Testament		Recep Tayyip Erdoğan Speeches		Mevlana Mesnevi	
Token	Count	Token	Count	Token	Count
,	5068	,	197068	,	37352
.	4523	.	154310	.	26495
’	2999	’	63672	bir	8670
“	1879	ve	53993	o	5785
”	1670	bir	50773	bu	5712
isa	1313	bu	47407	!	4521
o	883	de	26254	de	4489
bir	747	da	26029	da	3966
ve	731	için	15672	’	3818
dedi	666	türkiye	14280	ve	3594
?	639	her	12493	?	3387
bu	546	çok	12345	ki	3076
da	474	daha	12154	tanrı	2931
nın	465	?	11373	“	2875
!	455	biz	10916	”	2578
için	433	en	10582	ne	2204
de	429	ne	9610	gibi	2101
‘	386	olarak	9349	;	1798
diye	383	kadar	9334	dedi	1713
ne	368	tüm	9051	ey	1625

Table 4.1: Corpora Top Frequence Words

Bible - New Testament		Recep Tayyip Erdoğan Speeches		Mevlana Mesnevi	
Token	Count	Token	Count	Token	Count
isa	1313	türkiye	14634	tanrı	2941
dedi	670	büyük	6723	dedi	1724
tanrı	319	aynı	5750	ey	1631
adam	179	değerli	5274	yok	739
rab	176	yeni	5247	başka	732
oğlu	161	bugün	5176	can	665
verdi	159	ediyorum	4938	adam	644
dediler	149	birlikte	4664	eder	541
zaman	139	önemli	4335	kötü	501
birlikte	138	devam	4316	gelir	462
üzerine	138	son	4275	yüzünden	460
söyleyeyim	125	sadece	4197	vardır	455
baba	122	kardeşlerim	4110	su	447
in	120	terör	3729	geldi	432
petrus	114	zaman	3419	güzel	432
yahya	105	anda	3287	doğru	421
iman	103	allah	3261	yüzlerce	421
kutsal	101	ülkemizin	3206	gönül	420
büyük	98	İstanbul	3201	yol	417
yanına	96	özellikle	3190	meydana	407

Table 4.2: Top Frequence Words Without Punctuations and Stopwords

4.3. Text Vectorization

The process of vectorizing text is a crucial step in the development and implementation of deep learning models for tasks related to natural language processing like text generation. The goal is to convert unprocessed textual data into a numeric representation that the model can understand and handle. In this study, we used different text vectorization techniques below to compare the performance of the text generation models.

4.3.1. Character-Based Vectorization - One Hot Encoding

Character-based vectorization deals with text data at the granular level of individual characters. In this approach, each character in the Turkish alphabet, plus a few special characters, are represented as a unique vector. One Hot Encoding is a common method used for character-level vectorization. In One Hot Encoding, each character is represented as a binary vector of a specific length. The length of the vector equals the total number of unique characters in the text corpus. Each character vector is populated with zeros, except for a single '1' at the index that corresponds to the character it represents (Hancock et al., 2020). As shown in Table 4.3, our bible corpus one hot encoding vector length is 40, Recep Tayyip Erdoğan Speeches corpus one hot encoding vector length is 55 and the last one Mevlana Mesnevi corpus one hot encoding length is 51.

Corpus	Total Chars	Unique Chars	Vocab Size
Bible New Testament	390.626	45	40
Recep Tayyip Erdoğan Speeches	18.958.685	82	55
Mesnevi - Mevlana	2.371.540	68	51

Table 4.3: Corpus Character-Based Tokens

4.3.2. Word-Based Vectorization - Embedding Layer

In our methodology, the embedding layer plays a pivotal role in word and subword-based vectorization. During preprocessing, we converted our Turkish text corpus into an integer-encoded vocabulary. This encoded vocabulary then served as input to the embedding layer. The layer, in turn, cross-referenced these integers in an embedding lookup table, subsequently outputting the corresponding embedding vectors. The semantically similar words ended up having vectors that were close to each other in the embedding space (Mikolov et al., 2013). This property of the embedding layer provided us with a powerful tool to interpret word associations and semantic similarities within our Turkish text corpus. Words with similar meanings gravitated closer in the embedding space, offering a rich, context-sensitive analysis of the language data. This feature greatly aided in the effective generation of Turkish text by our model. Table 4.4 shows that the bible corpus with word-based vocab size is 10000, Recep Tayyip Erdoğan Speeches corpus word-based

vocab size is 100000 and the last one Mevlana Mesnevi corpus word-based vocab size is 50000.

Corpus	Word-Based Tokens	Unique Tokens	Vocab Size
Bible New Testament	72.738	10.125	10.000
Recep Tayyip Erdogan Speeches	2.913.088	134.662	100.000
Mevlana - Mesnevi	432.261	52.374	50.000

Table 4.4: Corpus Word-Based Tokens

4.3.3. Subword-Based Vectorization - Embedding Layer

Subword-based vectorization is a more granular approach than word-based vectorization, and it operates on subword units such as morphemes. This technique can help the model deal with the morphological complexity of languages like Turkish (Sennrich et al., 2016).

Similar to word-based vectorization, an Embedding Layer is also used for subword-based vectorization. However, instead of representing whole words, each subword or morpheme is assigned a unique vector in this embedding space. As a result, even if the model encounters an unseen word, it can handle it by breaking it down into known subwords or morphemes, providing a way to handle out-of-vocabulary words. This makes subword-based vectorization particularly effective for morphologically rich languages like Turkish (Devlin et al., 2019). We used byte-coded encoding algorithms to create subword-based tokens as shown in Table 4.5, bible corpus with subword-based vocab size is 9810, Recep Tayyip Erdoğan Speeches corpus subword-based vocab size is 24898, and the last one Mevlana Mesnevi corpus subword based vocab size is 25434.

Corpus	SubWord-Based Tokens	Unique Tokens	Vocab Size
Bible New Testament	76.420	9.810	9.810
Recep Tayyip Erdogan Speeches	3.373.166	24.898	24.898
Mevlana - Mesnevi	478.674	25.434	25.434

Table 4.5: Corpus SubWord-Based Tokens

4.4. Deep Learning Models

4.4.1. LSTM

In this thesis, we have created an LSTM neural network structure in different configurations to capture the performance variance depending on the utilized vectorization strategy. Specifically, we utilized an embedding layer for word-based and subword-based vectorizations, while for the character-based approach, we used a one-hot encoding scheme.

Each LSTM layer in our model is trained to recognize patterns across sequences, learning to capture the contextual dependencies in the input data. With two stacked LSTM layers, our network is designed to extract complex patterns in sequences effectively, an architecture made more effective by the memory characteristic of LSTM layers, which retain relevant past information to influence future predictions.

The output from the second LSTM layer feeds into a Dense layer, which essentially operates as a classifier on top of the features extracted by the LSTM layers. Given the multi class nature of our problem, predicting the next word or character from our vocabulary, we employed a softmax activation function in the Dense layer. If W_s and b_s represent the weights and bias of this layer, h_t represents the last LSTM hidden state, the probability distribution P_t is calculated as

$$P_t = softmax(W_s * h_t + b_s)$$

This function delivers a probability distribution over our target classes, guaranteeing that the sum of probabilities for all potential next words equals one.

During the model training process, we used the Adaptive Moment Estimation (ADAM) optimizer and categorical cross-entropy as the loss function. ADAM is an optimization algorithm that manages efficiently with sparse gradients on noisy problems, which aligns well with the challenges presented by text generation tasks (Kingma and Ba, 2014).

We chose categorical cross-entropy as our loss function given its appropriateness for multi-class classification problems. It measures the dissimilarity between the predicted and actual probability distributions for the target classes (Goodfellow et al., 2016).

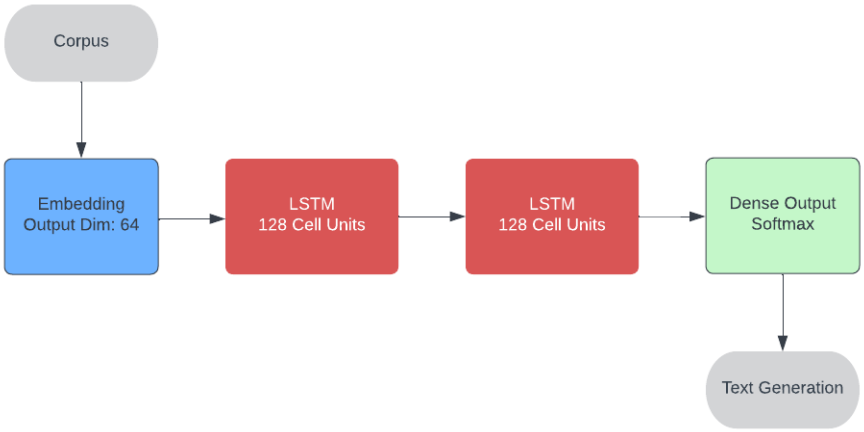

Figure 4.2: LSTM Model

4.4.2. Encoder - Decoder Model

Our model incorporates an embedded layer with both word-based and subword-based representations and a variant without an embedded layer using one-hot encoding.

The encoder component of our model comprises two LSTM layers. Using multiple LSTM layers enables the encoder to learn increasingly abstract representations of the input text. Each LSTM layer in the encoder receives sequential input data and processes it to generate a sequence of hidden states. The last LSTM layer's hidden state and cell state make up context vector of encoder side.

The decoder component takes the encoded information from the encoder and generates the output sequence word by word. It consists of a single LSTM layer followed by a dense layer with a softmax activation function. The LSTM layer in the decoder utilizes the encoded information to initialize its hidden state, allowing it to leverage the learned representations from the encoder.

The dense layer with a softmax activation function is responsible for producing the probability distribution over the vocabulary, indicating the likelihood of each word in the output sequence. During training, the decoder is optimized to generate sequences that maximize the likelihood of the ground truth target sequences.

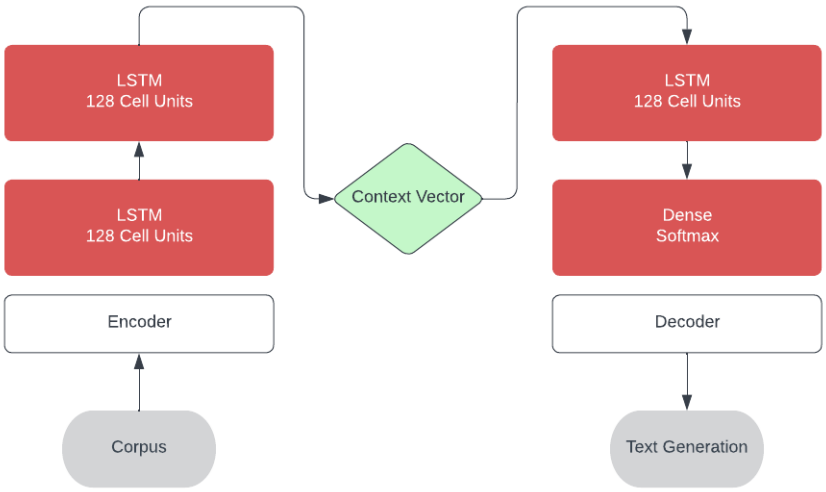

Figure 4.3: Encoder Decoder Model

4.4.3. GPT

The architecture of GPT is based on the ”decoder” portion of the original Transformer. This design choice was motivated by the need for a model that could generate a sequence of tokens one after another, which is crucial for the task of text generation.

The model consists of a stack of transformer decoders, with each decoder made up of two parts: a masked self-attention mechanism and a position-wise feed-forward network.

We implemented the GPT model using the Tensorflow Keras framework due to its dynamic computational graph and the active support from its community (Abadi et al., 2015). To feed our corpus data as Tensorflow data pipeline into the model, we used a Byte Pair Encoding (BPE) tokenizer, which tokenizes the text at the subword level, reducing the chance of encountering out-of-vocabulary words (Sennrich et al., 2016).

The model was trained with a language modeling objective, where the goal was to predict the next token in the sequence given the previous tokens. The model’s predictions were compared against the actual tokens to compute the loss using the Cross-Entropy Loss function. The Adam optimizer was then used to adjust the model’s parameters to minimize the loss.

The training process required careful tuning of hyperparameters, including the learning rate, batch size, and the number of training epochs. Additionally, due to the high computational demands of the GPT model, special attention was paid to memory management

during training to prevent out-of-memory errors.

4.4.4. GAN

In this study, standard GAN model could not be used due to the converge problem of structured sequence data. Text data naturally is structured with words and sentences as tokens in sequence forming coherent meaning. Text generation requires coherent grammar, semantics, and logical flow, which is challenging for GANs. To overcome converge problem, we implemented GAN with the Gumbel-softmax Distribution (GSGAN), particularly designed for sequence generation tasks, for text generation task. Similar to standard GANs, GSGAN includes a Generator and a Discriminator. The Generator's role is to produce synthetic sequences, while the Discriminator's task is to distinguish between real sequences from the dataset and synthetic ones generated by the Generator (Kusner and Hernández-Lobato, 2016).

We leveraged the Gumbel Softmax trick within the generator. Instead of generating one-hot vectors directly (which is non-differentiable), it allowed us to generate a differentiable approximation to one-hot vectors, enabling backpropagation of gradients from the discriminator to the generator.

The Gumbel-Softmax distribution is a method of drawing samples Z from a categorical distribution with k classes using a continuous distribution. This is particularly useful when differentiating through the sample is desired.

First, we sample some Gumbel noise, e. For a random variable $X\ Uniform(0, 1)$, the variable $e = -log(-log(X))$ follows a Gumbel distribution.

Second, we compute logits y, which are unnormalized log probabilities that sum to one. Given a vector of class probabilities $p = [p1, p2, ..., pk]$, the logits are $y = log(p) + e$.

Finally, the Gumbel-Softmax sample is using softmax function. Given a temperature parameter $\tau > 0$, the sample z:

$$z = softmax((1/\tau) * y)$$

The temperature parameter τ controls the sharpness of the sample. As τ approaches 0, the sample z becomes a one-hot encoded vector, picking out a single class with probability near 1. As τ approaches infinity, the sample z approaches a uniform distribution over

the k classes. In other words, lower temperatures make the distribution over classes more discrete, and higher temperatures make it more uniform (Kusner and Hernández-Lobato, 2016).

The generator was trained to fool the discriminator, using gradients to improve. The discriminator was trained to distinguish real text from generated text. The Gumbel Softmax function was applied to the output of the generator.

The Adam optimizer with 0.001 learning rate was utilized to adjust the weights in both the generator and discriminator networks.

4.5. Sampling Methods

In this thesis, four sampling techniques were used to generate Turkish text using deep learning models. These sampling techniques have unique strengths and potential that affect the quality, diversity, and novelty of the generated text. The purpose of this section is to elaborate on the procedures and principles behind each method.

In the context of this study, each of these sampling methods was implemented and evaluated based on the quality, coherence, diversity, and novelty of the generated Turkish text. By comparing the performances of different models utilizing these sampling techniques, we aimed to identify the most effective strategy for Turkish text generation with deep learning models.

4.5.1. Greedy Sampling

Greedy sampling, also known as maximum likelihood sampling, is the simplest form of sampling. In this approach, at each time step, the model selects the token with the highest probability in the next token prediction distribution. While this method is computationally efficient and often produces grammatically correct sentences, it has a tendency to generate repetitive and less diverse output. This is because the model always takes the safest path by choosing the highest probable word, thereby minimizing the possibility of novel text generation (Leblond et al., 2021).

4.5.2. Temperature Sampling

Temperature sampling is a kind of the greedy sampling method that adds a parameter named temperature, to modulate the probability distribution of the next token prediction. When the temperature is set to a value close to zero, it mimics greedy sampling and produces less diverse output. Contrarily, as the temperature value approaches one, it makes the distribution more uniform, thereby increasing diversity and creativity in the generated text. However, a higher temperature may also produce more unpredictable and potentially nonsensical output. Therefore, striking the right balance is essential to generate meaningful yet diverse text (Dhamala et al., 2022).

4.5.3. Top-k Sampling

Top-k sampling is a technique that adds more diversity to the model's output by narrowing down the sampling pool to the top-k most probable words. The model randomly selects the next word from this reduced pool instead of considering the whole vocabulary. This technique helps prevent highly improbable words from being selected and reduces the chances of generating nonsensical sentences. The value of k can be fine-tuned to balance between diversity and coherency of the generated text (Dhamala et al., 2022).

4.5.4. Beam Search

Beam search is an advanced sampling method that considers multiple paths simultaneously while generating text. At each step, it expands each of the current top-k paths, resulting in k^2 possible paths. From these, it selects the top-k paths based on their cumulative probabilities. Beam search is a balance between greedy sampling and exhaustive search, as it looks ahead to make the best decision but keeps its search space manageable by limiting the number of options considered (Leblond et al., 2021).

4.6. Evaulation Metrics

In this study we have used PPL and BLEU evaluation metrics for performance comparison of deep learning models. Although the BLEU metric is used to calculate machine

translation quality, we have utilized corpus-based BLEU along with BLEU-2, BLEU-3, and BLEU-4 scores.

5. RESULTS

We applied different sampling algorithms together with different tokenization methods to measure the performance of the models. It has been observed that the encoder decoder model gives better results with the small dataset, and the GPT model gives better results with the large dataset. We also saw that subword tokenization performance is higher than character-based and word-based tokenization.

The findings of my thesis comparing deep learning models for text generation in the Turkish language indicate that byte code encoding with subwording is the most suitable tokenization process for the agglutinative structure of Turkish. It is evident that word-level tokenization, due to its dependency on the corpus, is inadequate compared to other approaches for text generation. Even in basic models like LSTM, character-level tokenization has been observed to accurately capture the vowel harmony, punctuation marks, and the formation of affixes in the Turkish language.

Furthermore, it was observed that the size of the corpus is also crucial. In the case of the GPT model, it was noticed that utilizing a small-sized corpus resulted in the generation of sentences that merely imitated the existing corpus. However, when a larger corpus was employed, it became evident that the GPT model produced text that better adapted to the given context. The utilization of a larger corpus facilitated a more accurate representation of the linguistic patterns and semantic nuances present in the Turkish language, thereby enhancing the coherence and quality of the generated text.

Table 5.1. shows that in the Bible corpus encoder-decoder model with character-based tokenization is much more accurate while GAN model BLEU scores are high. The related generated text is:
prompt: ”bugün burada olmak için”
bugün burada olmak için sevinçle kalabalığı söyledikten sonra isa, “bu adam isa’nın ardından gitti. bu arada o’nu yaklaşıp

Table 5.2. shows that in the Mesnevi corpus has better performance on encoder-decoder model with character-based tokenization. The related generated text is:

prompt: "bugün burada olmak için"
bugün burada olmak için değil, yalnız ona doğru olmak için. çünkü o, ancak odur.

Table 5.3. shows that in the Recep Tayyip Erdoğan Speeches corpus has better performance on GPT model with subwording tokenization in all metric both PPL and BLEU's. The related generated text is:
prompt: "bugün burada olmak için"
bugün burada olmak için bir mücadele noktasında, çok daha fazla hassasiyet gösteriyor ve soruyorum, ey avrupa birliği, chp'ye gönül verenler, bu tezgahı hep birlikte şehit oldular. kardeşlerim, sizlerden ricam şudur: yalancıdan bize düşen milli

Model	Tokenization	Sampling	PPL	BLEU-2	BLEU-3	BLEU-4
LSTM	Char Based	Greedy	97	0.8660	0.7337	0.5940
LSTM	Word Based	Greedy	94	0.7071	0.2827	0.000
LSTM	Sub-Word Based	Greedy	87	0.8820	0.7660	0.6300
Encoder Decoder	Char Based	Greedy	**84**	0.8321	0.8222	0.5642
Encoder Decoder	Word Based	Greedy	88	0.6900	0.6879	0.0000
GPT	Sub-Word Based	Greedy	99	0.4907	0.2886	0.1460
GAN	Sub-Word Based	Greedy	110	**0.9621**	**0.7928**	**0.6436**

Table 5.1: Bible Corpus - Model and Granularity Comparison

Model	Tokenization	Sampling	PPL	BLEU-2	BLEU-3	BLEU-4
LSTM	Char Based	Greedy	96	0.7583	0.4254	0.0000
LSTM	Word Based	Greedy	95	0.6901	0.2913	0.000
LSTM	Sub-Word Based	Greedy	96	0.8820	**0.7660**	**0.6300**
Encoder Decoder	Char Based	Greedy	**90**	0.8624	0.5230	0.1420
Encoder Decoder	Word Based	Greedy	94	0.7212	0.4719	0.2015
GPT	Sub-Word Based	Greedy	93	0.8161	0.5959	0.0000
GAN	Sub-Word Based	Greedy	140	**0.8917**	0.6519	0.3448

Table 5.2: Mesnevi Corpus - Model and Granularity Comparison

Model	Tokenization	Sampling	PPL	BLEU-2	BLEU-3	BLEU-4
LSTM	Char Based	Greedy	96	0.8584	0.6292	0.4123
LSTM	Sub-Word Based	Greedy	99	0.7071	0.2827	0.0000
Encoder Decoder	Char Based	Greedy	90	0.8624	0.5230	0.1420
GPT	Sub-Word Based	Greedy	**89**	**0.9620**	**0.7524**	**0.5730**
GAN	Sub-Word Based	Greedy	450	0.4836	0.1678	0.0000

Table 5.3: Recep Tayyip Erdoğan Speeches Corpus - Model and Granularity Comparison

In sampling comparison Table 5.4 shows that low temperature and min k value performed well.

Ex.	Model	Token	Sampling	PPL	BLEU-2	BLEU-3	BLEU-4
*01	LSTM	Char	Top-K (k=1)	**89**	**0.8073**	**0.7192**	**0.6230**
*02	LSTM	Char	Top-K (k=3)	95	0.3904	0.2557	0.1728
*03	LSTM	Char	Top-K (k=5)	91	0.6469	0.4886	0.3840
*04	LSTM	Char	Top-K (k=7)	92	0.5550	0.3863	0.2272
*05	LSTM	Char	Top-K (k=9)	94	0.5477	0.3774	0.2207
*06	LSTM	Char	Top-K (k=15)	95	0.5158	0.3188	0.2103
*07	LSTM	Char	Top-K (k=20)	93	0.6019	0.4075	0.2814
*08	LSTM	Char	Temp. (temp=0.1)	**87**	0.7207	0.5697	0.4116
*09	LSTM	Char	Temp. (temp=0.2)	88	**0.7948**	**0.6919**	**0.5635**
*10	LSTM	Char	Temp. (temp=0.3)	92	0.6726	0.4954	0.3391
*11	LSTM	Char	Temp. (temp=0.4)	90	0.7071	0.5452	0.4412
*12	LSTM	Char	Temp. (temp=0.5)	91	0.7345	0.6168	0.5132
*13	LSTM	Char	Temp. (temp=0.6)	97	0.5620	0.3785	0.2359
*14	LSTM	Char	Temp. (temp=0.7)	92	0.7054	0.5430	0.4323
*15	LSTM	Char	Temp. (temp=0.8)	99	0.5239	0.3286	0.2187
*16	LSTM	Char	Temp. (temp=0.9)	92	0.7439	0.5982	0.4217
*17	LSTM	Char	Temp. (temp=1.0)	98	0.5298	0.4391	0.3187

Table 5.4: Bible Corpus - Sampling Comparison

- **01* gece yarısı ona gidip kendisine dokunanların gelenlerini bilenli ve yahudiler'in kralı'ya dedi ki, "bu adam isa'nın yanın

- **02* gece yarısı ona gidip kadar daha doğru gitti. yerleklerin ve kendisiyleri yok ve yanından daha gelmek istemiştir. size şö
- **03* gece yarısı ona gidip söyledim. beni de onlara şöyle dedi: "işte onlara iki gibi yıkılmış öbür, ama iki öğlere çağrılacak
- **04* gece yarısı ona gidip buluyordu. çünkü götülor. rab'bin süzgün verirseniz, böylece günüşte ve ona şikremekten babam'ın ke
- **05* gece yarısı ona gidip kadınları koyununa kay beytilecek ver. çüncü şeylerin ininsem! o'nun izin verdi. o da gelip olup be
- **06* gece yarısı ona gidip kendilerine hazırlamayacak ortuluruştu. herkes tamın oğlu, hiç kimseyle keye söylelerle yok ve tarr
- **07* gece yarısı ona gidip onlar tanılarımı görmek için tutulkun isa'yı yok diye, şemeni de gözdür. yoksa bütün benden bir sek
- **08* gece yarısı ona gidip kendisine dokunanların görüyorum. beni sen de beni göndereni yanına geldi. isa onlara, "size doğrus
- **09* gece yarısı ona gidip kendisine dokuna dönüp o'nun ardından gitti. bu arada o'nun için değil mi? bunlar isa'nın çok ve gö
- **10* gece yarısı ona gidip kadınların önünde yere ve öldürmek istemeye başladı. "sizi halka kapanınızı aldı, onlara şöyle dedi
- **11* gece yarısı ona gidip kötü ruhu etmedi. onlara şöyle dedi: "bu adam önce cin tepedeceklerini yok acıdan ve gölde ve adam
- **12* gece yarısı ona gidip de ayaklarına görüyordu. başka bir tek taraf evinde yerine gelsin diye oldu: "insanoğlu'nun senden
- **13* gece yarısı ona gidip kızka içinde bir sırada yağ düşününsa, bu kadın sevinin başına gelip onlara ayarlamaya görünmüşan v
- **14* gece yarısı ona gidip kente gittiğinde olan bir şey zamacağım yitiremek için halkı olmasını istiyorsun?" diye sordu. isa
- **15* gece yarısı ona gidip koyunlarınızı işitince yeryüse uyanına gidiyordu. bir kâse horoz öteki gelenlerinin başına, çocukla

- **16* gece yarısı ona gidip kendisini severlerdi. başkâhinler, din bilginlerinin adasa, “isa’yı öğrencilerine, “bu adam kapısın
- **17* gece yarısı ona gidip tefiz giymileri görmem meseriyle ileri gelenler, isa’ya alınacağı yörünekleşmiş rabut ucundan atard

5.0.1. Genereted Text Examples

- *Model: Encoder Decoder Sampling: Greedy Corpus: Bible*
  yapay zeka ile ilgili soru sordular. isa onlara, “neden ne?” isa konuşmasını aralarında, isa’ nın öğrencileri o’ na
- *Model: Encoder Decoder Sampling: Temperature (0.3) Greedy Corpus: Bible*
  yapay zeka ile ilgili soru sordular. isa, “size söyledim, onu” dedi. adam da adamlardan isa’ nın ardından giden kalabalıktan
- *Model: Encoder Decoder Sampling: K-Sampling (k=20) Greedy Corpus: Bible*
  yapay zeka ile ilgili soru sordular. isa, “gelin” dedi, bunun üzerine şu yaklaştı diye görünce kalabalık o’ nun ardından yere
- *Model: Encoder Decoder Sampling: Greedy Corpus: Recep Tayyip Erdoğan Speeches*
  yapay zeka ile ilgili soru sordular. vatandaşlarımız diyerek her [UNK] gibi iyi alıyoruz. sizler dünyasını hizmet razı ettik . [UNK] göre [UNK] geldik , ve
- *Model: Encoder Decoder Sampling: Temperature (0.3) Corpus: Recep Tayyip Erdoğan Speeches*
  yapay zeka ile ilgili soru sordular. sık avrupa vesilesiyle, ulaşım yatırımlarından anadolu göre daha [UNK]
- *Model: Encoder Decoder Sampling: K-Sampling (k=20) Corpus: Recep Tayyip Erdoğan Speeches*
  yapay zeka ile ilgili soru sordular. vatandaşlarımız diyerek her şehirde olacak ne de devam eden üniversite
- *Model: Encoder Decoder Sampling: Greedy Corpus: Mesnevi*
  yapay zeka ile ilgili soru sordular. İşte bu O’ dur. dedi Ne de âlemi, [UNK] Nihayet Şimdi, [UNK] yüzlerce, orada ancak ki

- *Model: Encoder Decoder Sampling: Temperature (0.3) Corpus: Mesnevi*
  yapay zeka ile ilgili soru sordular. O, kendi birlik Tanrıdan gözü gör eğer, koku dök sen eğer. [UNK] serkeş çıkar da ağlar!

- *Model: Encoder Decoder Sampling: K-Sampling (k=20) Corpus: Mesnevi*
  yapay zeka ile ilgili soru sordular. O, ey; yok mu de ya… dedi, şimdi yine delildir yoktur… geldi. bulursun kökü

- *Model: GSGAN Converge: Gumbel Softmax Corpus: Recep Tayyip Erdoğan Speeches*
  ben başbakanlığım dönemimde yine de yine tekrar ifade ediyor , ondan sonra bu [UNK] . cumhurbaşkanı nasıl [UNK] , bu tür temasları [UNK] gerekiyor . cezayir'in milli çizgide duran

- *Model: GSGAN Converge: Gumbel Softmax Corpus: Recep Tayyip Erdoğan Speeches*
  bugün türkiye'nin 2023 hedeflerimiz doğrultusunda sürdürdüğü kararlı ve heyecan verici hikâyeleriyle doludur .

- *Model: GSGAN Converge: Gumbel Softmax Corpus: Recep Tayyip Erdoğan Speeches*
  bugüne kadar allah'ın izniyle bizi destekledi; süreç halen bitmiş değil. süreç tamamlanmış durumda. şimdi sizlere, türkiye, şimdi, 10 yılı , 10 milyar dolar olan

- *Model: GSGAN Converge: Gumbel Softmax Corpus: Recep Tayyip Erdoğan Speeches*
  bugün büyük bir hesap vermek zorunda kaldıkları [UNK] her gün birlik ve beraberlik içinde olduk . dayanışma içinde yurtlarına dönebilecekleri eman bölgeleri haline getireceğiz . arap halkları , hem bölgedeki kürt hem bölgedeki kürt olmaktan çıkmış haline gelmiştir . bunların asıl

- *Model: GSGAN Converge: Gumbel Softmax Corpus: Recep Tayyip Erdoğan Speeches*
  böylesi müstesna bir atmosferi teneffüs edebiliyorsak , önce allah'ın birliği ve dört elle sonra da hep birlikte ülkemize yüksek düzeyli stratejik işbirliği koordinasyon kurmanın çabası içindeyiz . geçtiğimiz 150 , [UNK] ve refah temasının birliği değerlerini

- *Model: GSGAN Converge: Gumbel Softmax Corpus: Recep Tayyip Erdoğan Speeches*
  türkiye büyük hedefleri doğrultusunda kararlılıkla sürdüreceğiz . dünya nüfusunun yarısından fazlasının elektriğe , o kadar geçişler başladığı gündür , aile ve sabrı , aile ve gazilerimize sağlık ve [UNK] .

- *Model: GPT*

  sevgili kardeşlerim şimdi sizlere sesleniyorum; bu millet, bu noktada şu anda 2 yıl önce sayın başbakan, bu noktada çok çok açık, değerli basın mensupları, bunu ifade ediyor. değerli muhtar kardeşlerim, ama ben buradan bir şey, ben sizlere

- *Model: GPT*

  sevgili kardeşlerim şimdi sizlere sevgilerimi gönderiyorum. sevgili gençler, kıymetli kardeşlerim, hanımefendiler, beyefendiler; sizleri en kalbi muhabbetlerimle selamlıyorum. bugün de, muhabbetle selamlıyorum. inşallah bu gömleğe sığmıyor. türkiye olarak, bugün de yeni anayasasına, yeni bir gençlik

- *Model: GPT*

  sevgili kardeşlerim şimdi sizlere ben inanıyorum, aramızda olan gençlik değildi, 19 mayıs ruhu yerine 23 nisan imam hatip ortaokulu var. imam hatip açıyoruz, van, bu okulları kapatacağız diyerek çıkıyorlar. bunu açıkça söylemiyorlar.

6. CONCLUSION

In our research, we observed that subword tokenization, particularly tailored for the Turkish language, outperformed other methods. Our findings suggest that subword tokenization is best suited for the agglutinative nature of Turkish. Furthermore, for evaluating the performance of different models, considering machine-based metrics such as bigram perplexity and BLEU scores (including BLEU-2, BLEU-3, and BLEU-4) proved to be a comprehensive approach.

Among the models tested, GPT demonstrated superior semantic results when supplied with sufficient data. Additionally, the transformer architecture and attention mechanism exhibited promising results in generating Turkish-specific text compared to other models.

The compatibility of subword embeddings in the Transformer structure, along with the attention mechanism and positional embeddings, aligns well with the structure of the Turkish language. With the expanding utilization of pre-trained models based on transformer structures for text generation in Turkish, we anticipate achieving significantly improved results.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Bahdanau, D., Cho, K. H., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Brauwers, G. and Frasincar, F. (2023). A general survey on attention mechanisms in deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3279–3298.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ..., and Amodei, D. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Camacho-Collados, J. and Pilehvar, M. T. (2018). On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis.

Celikyilmaz, A., Clark, E., and Gao, J. (2020). Evaluation of text generation: A survey.

Chakraborty, S., Banik, J., Addhya, S., and Chatterjee, D. (2020). Study of dependency on number of lstm units for character based text generation models.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2015). Gated feedback recurrent neural networks. In *International conference on machine learning*, pages 2067–2075. PMLR.

de Rosa, G. H. and Papa, J. P. (2021). A survey on text generation using generative adversarial networks. *Pattern Recognition*, 119:108098.

Demir, S. and Oktem, S. (2023). A benchmark dataset for turkish data-to-text generation. *Computer Speech & Language*, 77:101433.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhamala, J., Kumar, V., Gupta, R., Chang, K.-W., and Galstyan, A. (2022). An analysis of the effects of decoding algorithms on fairness in open-ended language generation.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Fatima, N., Imran, A. S., Kastrati, Z., Daudpota, S. M., and Soomro, A. (2022). A systematic literature review on text generation using deep neural network models. *IEEE Access*, 10:53490–53503.

Fedus, W., Goodfellow, I., and Dai, A. M. (2018). Maskgan: better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*.

Gatt, A. and Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

Graves, A. (2014). Generating sequences with recurrent neural networks.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA. Association for Computing Machinery.

Gu, J., Wu, Q., and Yu, Z. (2020). Perception score, a learned metric for open-ended text generation evaluation. *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, 14B:12902–12910.

Hancock, J. T., Khoshgoftaar, T. M., and Hancock, K. J. (2020). Survey on categorical data for neural networks. *Journal of Big Data*, 7:28.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9:1735–1780.

Iqbal, T. and Qureshi, S. (2022). The survey: Text generation models in deep learning. *Journal of King Saud University - Computer and Information Sciences*, 34:2515–2528.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2022). Auto-encoding variational bayes.

Kusner, M. J. and Hernández-Lobato, J. M. (2016). Gans for sequences of discrete elements with the gumbel-softmax distribution. In *Advances in Neural Information Processing Systems*, pages 4686–4694.

Kutlugün, M. A. and Şirin, Y. (2018). Turkish meaningful text generation with class based n-gram model. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4.

Leblond, R., Alayrac, J.-B., Sifre, L., Pislar, M., Lespiau, J.-B., Antonoglou, I., Simonyan, K., and Vinyals, O. (2021). Machine translation decoding beyond beam search.

Lecun, Y. (2001). A theoretical framework for back-propagation.

Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature 2015 521:7553*, 521:436–444.

Li, Z., Jiang, X., Shang, L., and Li, H. (2018). Paraphrase generation with deep reinforcement learning.

Lin, K., Li, D., He, X., Zhang, Z., and Sun, M.-T. (2017). Adversarial ranking for language generation. *Advances in neural information processing systems*, 30.

Luo, T., Cao, X., Li, J., Dong, K., Zhang, R., and Wei, X. (2021). Multi-task prediction model based on ConvLSTM and encoder-decoder. *Intelligent Data Analysis*, 25(2):359–382.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

Mielke, S. J., Alyafeai, Z., Salesky, E., Raffel, C., Dey, M., Gallé, M., Raja, A., Si, C., Lee, W. Y., Sagot, B., and Tan, S. (2021). Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Montesinos, D. M. (2020). Modern methods for text generation.

Niu, Z., Zhong, G., and Yu, H. (2021). A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62.

OpenAI (2023). Gpt-4 technical report.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Peng, B. and Yao, K. (2015). Recurrent neural networks with external memory for language understanding.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Puduppully, R., Dong, L., and Lapata, M. (2019). Data-to-text generation with content selection and planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6908–6915.

Radford, A. and Narasimhan, K. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for squad.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408.

Santhanam, S. (2020). Context based text-generation using lstm networks.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Stahlberg, F. (2020). Neural Machine Translation: A Review | Journal of Artificial Intelligence Research. https://jair.org/index.php/jair/article/view/12007.

Suta, P., Lan, X., Wu, B., Mongkolnam, P., and Chan, J. H. (2020). An overview of machine learning in chatbots.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-December:5999–6009.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M.,

and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation.

Xie, S. M., Santurkar, S., Ma, T., and Liang, P. (2023). Data selection for language models via importance resampling.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Zhang, Y., Gan, Z., Fan, K., Chen, Z., Henao, R., Shen, D., and Carin, L. (2017). Adversarial feature matching for text generation. In *International Conference on Machine Learning*, pages 3881–3890.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537.

Zurada, J. (1992). *Introduction to artificial neural systems*. West Publishing Co.

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı	**Murat Güzel**

Eğitim

Ahmet Yesevi Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği
Anadolu Üniversitesi Yönetim Bilişim Sistemleri
İstanbul Üniversitesi Felsefe
Atatürk Üniversitesi Sosyoloji

Yazar, şu anda kurucuları arasında yer aldığı özel bir yazılım şirketinde mesleki kariyerine devam etmektedir.