

**Examining Cause-Effect Relationships with Large Language Models**

Submitted by:

*Zeynep Sude İleri*

*Ayşenur Özbek*

*Sema Balkaya*

*Umut Kılınç*

*Tevfik Eren Yavuz*

Asst. Prof. Orhan Deniz Gençağa

Faculty of Engineering

Antalya Bilim University

2025

**CONTENTS**

**4. DATA PREPARATION** ...................................................................................... 17  
4.1 **Tokenization** ................................................................................................. 18  
4.1.1 Splitting Text into Tokens ............................................................. 18  
4.1.2 Byte Pair Encoding (BPE) and its Role in Tokenization ................. 18  
4.1.3 **Embeddings** ........................................................................................ 19

**7. TRAINING AND FINE-TUNING** ......................................................................... 29  
7.1 **Pretraining** ................................................................................................... 30  
7.1.1 Next-Word Prediction Tasks ...................................................... 30  
7.1.2 Importance of Self-Supervised Learning ................................... 30  
7.2 **Fine-Tuning** Approaches ............................................................................. 31  
7.2.1 Classification Tasks ................................................................. 31  
7.2.2 Instruction Fine-Tuning ............................................................. 31  
7.2.3 Integration of Labeled Datasets ................................................. 32

**5. ATTENTION MECHANISMS** ............................................................................ 21  
5.1 **Self-Attention** ............................................................................................... 22  
5.1.1 Capturing Dependencies Within Sequences ................................ 22

**3. KEY CONCEPTS IN LLMS** ................................................................................ 13  
3.1 **Transformer Architecture** ............................................................................. 14  
3.1.1 Encoder-Decoder Structure .......................................................... 14  
3.1.2 **Self-Attention** Mechanism and its Importance .............................. 14

**8. APPENDIX A** ............................................................................................... 34

**REFERENCES**

**Introduction**

In recent years, the field of deep learning has made significant strides, particularly with the advent of advanced models in **Natural Language Processing (NLP)**, **Computer Vision**, and other domains. These models rely on powerful mechanisms and architectures that enable them to process vast amounts of data and perform complex tasks with high accuracy. One of the key components driving the success of these models is the combination of **tokenization**, **embedding**, **pretraining**, and **fine-tuning** techniques. These methods play a crucial role in optimizing the learning process, allowing models to generalize from large datasets and fine-tune for specific tasks with minimal data.

Tokenization, in its simplest definition, is a method that allows you to perform operations on data more easily by dividing data into smaller components. Embedding is basically the representation of data pieces, that is, tokens, in a vector space. Using a combination of pre-training and fine-tuning, deep learning models offer a powerful solution in terms of both efficiency and accuracy. During pre-training of the model, generalized information is learned over a large data set, while fine-tuning is optimized for a target task of that feature. For example, MobileNet V2 learns broad visual features before training and then fine-tunes this information to make it more specific in a task such as cat-dog handling. As another example, the BERT model provides high accuracy by acquiring general language knowledge in the pre-training process in language processing tasks, while optimizing this information during the fine-tuning process in tasks such as sentiment analysis [2].

By deploying these two methods together, it is possible to increase the model in both general and specific tasks and achieve higher accuracy with less data. Self-attention is a groundbreaking mechanism in deep learning that has transformed how models handle sequential data. It enables models to dynamically focus on the most relevant parts of an input sequence, regardless of their position, making it especially effective for tasks involving long-range dependencies. The mechanism plays a central role in modern architectures such as Transformers, which power state-of-the-art applications in natural language processing, computer vision, and beyond.

This report delves into the definition, purpose, and workings of the self-attention mechanism, highlighting its significance in deep learning advancements. Transformer architecture was introduced in 2017 as a revolutionary innovation in the field of sequential data processing. Thanks to its parallel processing capability and self-attention mechanism, it shows high performance in many tasks such as natural language processing, text translation, and text summarization. This architecture has become one of the cornerstones of artificial intelligence applications today.

**WHAT IS TOKANIZATION?**

Tokanization is, in its simplest terms, a method that allowsyou to perform operations on data more easily by separatingdata into smaller components. If we examine more deeply how Tokanization works and what it does; Tokanization is theprocess of separating data (for example, this can be text data or audio signal data) into smaller and more meaningful pieces. This process allows us to perform data analysis and processdata in many areas. The basic technique and logic behindTokanization are quite interesting because this process is oneof the basic steps of the Natural Language Processing (NLP) data processing process. Now let's examine this process andits function in more detail.

The basic principle of tokanization: Tokanization is theprocess of separating large data sets into smaller, moremanageable and functional pieces. Basically, raw data (forexample, words in a text file or time periods in a soundrecording) is separated into meaningful pieces and thesepieces are used for later use. These units are called 'tokens'. For example, if we consider a text data, tokens can be words, sentences, letters or word parts. The tokanizatiion processmakes it easier to perform later operations such as extractingmeaning, understanding the context, classifying or embedingby separating the text or any data. Tokanization can be considered as the process of separating a data into smaller, logical pieces in the most basic terms. Tokanization can be diversified according to the data set used, for example, if weare working on text data, tokanization for text data, if we areworking on numerical series, tokanization for numerical data, if we are working on image data, tokanization for image data or tokanization for audio data on audio data. We can do it.

If we consider tokenization for audio data, which is exactlywhat we focus on in the scope of our project, we can considerit more broadly;

audio data is usually divided into frequencies or time periodsin an audio recording. Audio data tokenization allows theaudio recording to be separated into meaningful units. Forexample, in speech recognition systems, sound waves areseparated into a series of frequency bands, and thesefrequencies are then converted into spoken words. Signal datacan usually include audio, radar, image processing, biomedicalsignals, or other types of analog/digital signals. Tokenizationof such data involves separating the meaningful parts of thesignal. Let's say we have an audio recording and we want toseparate this audio recording into tokens. The audio recordingis usually represented as a time series signal; the frequencyand amplitude of the sound waves change suddenly everysecond or smaller. The tokenization process in such data can be different, such as time slice separation, frequency featureseparation and Word or Phoneme Based Tokenization;

Time Slice Separation: Each time slice in the audio recordingcan be a token. This means dividing the audio recording intosmall pieces (for example, 1 second or 100ms each). In thisway, you can analyze each slice. If there is a speech in youraudio recording, these slices correspond to certain words orsound units. Frequency Feature Separation: The sound wavecan be separated into frequency components. By separatingthe audio recording into frequency components with Fouriertransform (FFT), you can consider each frequency band as a token. For example, each slice of the audio recording can havelow frequency (20 Hz - 200 Hz) and high frequency (2000 Hz - 5000 Hz) components. Each frequency component or bandcan be a token.

What is the Use of Tokenization of Signal Data?

Tokenization of signal data usually makes the data moremeaningful for further processing. For example, Making Data Manageable, Feature Extraction, More Effective Analysis andModeling, Preparation for Time Series Models are some of them. Tokenization of signal data is very important, especiallyfor audio or time series data. This process allows the signal tobe separated into meaningful parts so that the features of thesignal can be analyzed more efficiently. Tokenization is an important step for signal processing, speech recognition, sentiment analysis and many other applications.

metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**FIGURE-1**

This figure illustrates the sequence of **tokenization** and **embedding processes**, where raw text is first tokenized into smaller units and then converted into numerical vectors for further processing.

**WHAT IS EMMBEDINGS?**

Embeding is basically the representation of data pieces, namely tokens, in a vector space. Usually, we need numericalrepresentations for this data. These components, which wedivide our tokens, namely our data, into tiny pieces, can be words, sentences, text pieces, or even sound waves, images, and other data. The purpose of emmbeding is to transformthese elements into meaningful vectors that can be processedand analyzed more easily. These vectors are located in a multi-dimensional space and represent the mathematicalrelationship between these components. Depending on thetype of data set we use, emmbedding can be done on text data, Embedding on Image Data, Embedding on Audio Data, Embedding on Time Series Data. The embedding process is usually performed with deep learning models andunsupervised learning techniques. These techniques aretrained with very large datasets to learn the meaning of thedata, and the resulting embedding vectors represent thestructure of the data. Another thing that makes embeddingimportant is that it plays a very important role in the model'slearning of the context. For example, if we consider text data, it allows a word or element to learn meaning in context bytaking into account not only itself but also other elementsaround it. This contextual learning is carried out with modelssuch as transformer models. Embedding Embedding is one of the basic building blocks of deep learning models. Thesemodels try to learn the meaning of the data directly byperforming automated feature extraction. Embedding is a powerful way to capture the meaning of words in a numericalway. In this way, deep learning and machine learningalgorithms can be optimized to process words with numbers.

Embedding Methods:

Some popular embedding methods used after tokenization are: Word2Vec (Word to Vec), GloVe (Global Vectors for Word Representation), FastText, BERT (Bidirectional Encoder Representations from Transformers)

Let's go through a simple example to understand how tokenization and embedding work together in a languagemodel. It would be very meaningful to explain how embedding works through the concepts of "King", "Queen" and "Gender" because embedding learns the semanticrelationships between words in a numerical way. Suchrelationships include both similarities and differences in meaning between words. First, let's examine how embeddingtechniques represent words through numerical vectors andhow these vectors learn the relationship between words suchas "king" and "queen".

Word embeddings try to understand the meaningfulrelationship between words by representing this relationship in space (a mathematical vector space). In other words, wordswith similar meanings such as "king" and "queen" arerepresented by vectors that are close to each other in theembedding space.

Gender Context: The difference between King and Queen is essentially related to gender difference. When embeddinglearns such contexts, the word "king" means male leadershipand the word "queen" means female leadership. Therelationship between these words is a gender difference. If theembedding is trained correctly, the difference between "king" and "queen" can be represented by the distance between thevectors, which reflects a certain aspect, such as genderdifference.

In the embedding space, the vectors of the words "king" and"queen" are usually located in close proximity, while thedifference between them can be described as a genderdifference. If the embedding is trained correctly, we can express this difference by the difference between the vectors.

Tokenization is the process of breaking data (especially text) into meaningful units (tokens). This process is very importantwhen working with language data because it allows yourmodel to understand the text.

Embedding converts tokens into numerical vectors, allowingthem to learn the meaning of words. Embeddings are a fundamental building block for many applications in languageprocessing because the meaningful relationships betweenwords are located in the embedding space.

Tokenization and embedding work together to enablelanguage models to learn more efficiently and accurately. While tokenization breaks data into small pieces, embeddingconverts these pieces into a meaningful numericalrepresentation.

Audio Data and Embedding

When you work similarly to audio data embedding, you can analyze the frequencies, amplitudes, and time periods in theaudio recording. Audio embedding mostly works on thedigital representations of sound waves and tries to learn thesemantic features of this data.

In general, the following steps are followed for audioembedding:

Splitting the audio data into time periods: The audio recordingis usually divided into small time periods (for example, 20ms).

Frequency feature extraction: Separating the audio intofrequency components using techniques such as Fouriertransform.

Feature extraction: Using techniques such as Mel-frequencycepstral coefficients (MFCC) or spectrogramsextracting moremeaningful representations of the sound.

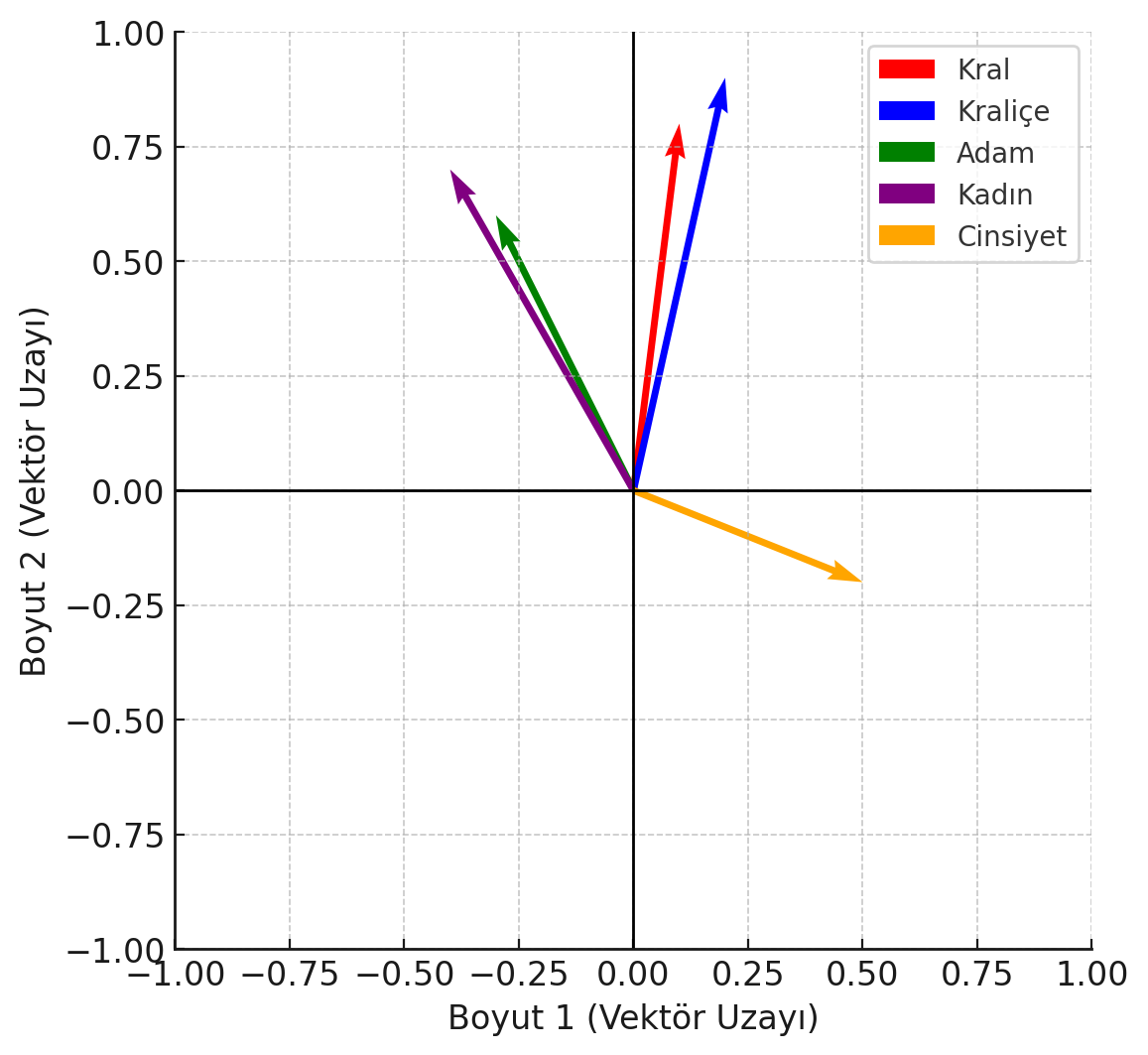
Creating embedding: Converting these features into numericalvectors using a model that can learn the semantic structure of the sound.

okenization breaks the data into small pieces so the model can process them.

Embedding represents these small pieces as numerical vectors, enabling mathematical learning of the meaning of thelanguage.

Token IDs are a tool that converts tokens into numerical form and are necessary for embedding to work successfully.

Tokenization and embedding are two critical steps thatcomplement each other in the process of learning the meaningof the language.



metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Pre-Training**

Pre-training is the process that allows deep learning models to learn general information by training them on a large data set. This process allows the model to essentially learn general properties and can be extended to different application domains. For example, visual modeling is trained using large data sets such as ImageNet [1]. In this process, the model learns basic visual features and can then use this information in different tasks. The biggest advantage of this method is that, through large data sets, the model has a wider information base and shows high performance with less data. Popular models such as BERT, GPT-3, and MobileNet V2 have achieved significant success by re-using general features learned through pre-training in target tasks [2]. However, this process is quite costly and time-consuming because high computing power is required to train the model with large data sets.

**Fine-Tuning**

Fine-tuning is the process of optimizing a pre-trained model for a target task. This process allows general features learned during pre-training to be focused on a specific data set. Fine-tuning usually involves retraining the model on the target task, allowing the model to obtain more accurate results on the target task. For example, while the BERT model learns general language features during the pre-training process, it can be optimized for specific emotion classifications by fine-tuning in an emotion analysis task [2]. Similarly, MobileNet V2, used in visual classification, can be retrained for a task such as cat-dog classification. Fine-tuning requires less data and provides high accuracy quickly. However, a careful optimization process is required to avoid overfitting the model while fine-tuning it.

**Relation Between Pre-Training and Fine-Tuning**

Using a combination of pre-training and fine-tuning, deep learning models offer a powerful solution in terms of both efficiency and accuracy. During pre-training of the model, generalized information is learned over a large data set, while fine-tuning is optimized for a target task of that feature. For example, MobileNet V2 learns broad visual features before training and then fine-tunes this information to make it more specific in a task such as cat-dog handling. As another example, the BERT model provides high accuracy by acquiring general language knowledge in the pre-training process in language processing tasks, while optimizing this information during the fine-tuning process in tasks such as sentiment analysis [2]. By deploying these two methods together, it is possible to increase the model in both general and specific tasks and achieve higher accuracy with less data.

****

**FIGURE-2**

This figure illustrates the sequential relationship between **Pre-Training** on a large dataset and **Fine-Tuning** on a specific dataset, facilitated through transfer learning.

**Self-Attention Mechanism**

**Self-Attention: Definition and Purpose**

Self-attention is a mechanism that enables models to dynamically focus on different parts of an input sequence while processing it. By learning the relationships between all tokens in the sequence, it provides a context-aware representation of each token. This capability is at the core of modern Transformer architectures, such as BERT and GPT, and has revolutionized tasks like machine translation, text summarization, and language understanding.

The self-attention mechanism computes a weighted representation of all input tokens for each token, allowing the model to determine which tokens are most relevant in a given context.

**How Self-Attention Works**

Self-attention involves the following computational steps:

1. **Input Representation**: Each input token is represented as a fixed-dimensional vector. For example, in a sentence, "I love cats," each word is mapped into an embedding vector of dimension .

2. **Query, Key, and Value Matrices**: For each token, three vectors are computed:

o **Query (Q):** Determines what information the token is looking for.

o **Key (K):** Represents the token's features.

o **Value (V):** Holds the token's information to be shared.

These are calculated as:

where are learnable weight matrices, and is the input token matrix.

3. **Attention Score Calculation**: Attention scores are computed by taking the dot product of and , capturing the similarity between tokens:

To stabilize training, these scores are scaled:

where is the dimensionality of the Key vectors.

4. **Softmax Normalization**: The scores are passed through a Softmax function to produce probabilities, ensuring that attention is distributed across all tokens:

5. **Weighted Sum**: The output for each token is computed as the weighted sum of the Value vectors:

**WHAT IS TRANSFORMER ARCHITECTURE?**

Transformer architecture is a neural network model developed by Vaswani and his colleagues in 2017 to provide solutions to challenges in natural language processing (NLP) and sequential data processing. Introduced in the article titled “Attention Is All You Need”, this architecture has revolutionized many applications such as language models and translation systems. Transformer overcomes the limitations of previous methods such as RNN (Recurrent Neural Network) and LSTM (Long Short-Term Memory) models, offering significant advantages in understanding the context in long texts and in computational efficiency by processing data in parallel. Since its development, it has become one of the most widely used approaches in text translation, summarization, text generation and many other NLP tasks. The main success of this model stems from its effective understanding of the relationship between all elements in the input, centered on a mechanism called self-attention. Transformer’s innovative structure is not limited to natural language processing alone, but has also become usable in different areas such as image processing (Vision Transformer).

Transformer architecture consists of two main components: encoder, which processes an input sequence, and decoder, which produces an output sequence from this input. Encoder, which transforms the input into a meaningful representation, transfers this representation to the decoder and produces the output. In addition to showing high performance in sequential data processing tasks, this structure is much faster than previous methods thanks to its parallel processing capability.

Encoder uses self-attention mechanism to process words in the text. This mechanism allows it to learn the context of a word with other words in the sentence. For example, in the sentence “The cat climbed the tree because it was very scared.”, it is understood from the context that the word “was scared” is related to “cat”. Transformer determines these relationships with self-attention and calculates how much each word is related to other words with attention scores. Thus, the model focuses more on important words.Encoder also uses positional encoding to understand the order of words. Because Transformer cannot directly learn the order of words while performing parallel processing. Positional coding allows the model to learn sequential dependencies by adding position information in the sentence to each word.

The solver produces the output sequentially using the representation from the encoder. During output estimation, the solver takes into account both the information it receives from the encoder and the previous words it produces. For example, during translation from Turkish to English, for the sentence “I read the book”, the solver first predicts the word “I” and then produces the words “read” and “the book” in order by making sense of the context.

One of the most powerful features of the Transformer architecture is that it combines the self-attention mechanism with multi-head attention. This allows the model to learn different contexts at the same time. For example, it can evaluate both the meaning of a word and its function in the sentence at the same time. This feature prevents context loss in long texts and helps the model develop a multi-faceted understanding.

As a result, the Transformer architecture has revolutionized natural language processing and sequential data processing with its parallel processing, long-range context understanding, and efficient learning. Today, it has become a standard method for many tasks such as text translation, text summarization, language modeling, and image processing.

The Transformer architecture has been a turning point in natural language processing and other sequential data processing fields. Thanks to its self-attention mechanism, parallel processing capability, and long context learning capacity, it has overcome the limitations of previous models and has formed the basis of modern artificial intelligence systems. Today, advanced technology models such as BERT, GPT, T5, and Vision Transformer are built on the Transformer architecture and are used in a wide variety of tasks such as language translation, text summarization, text generation, and even image processing.

The flexibility of this architecture has not only been limited to natural language processing, but has also enabled it to be successfully applied to different fields such as bioinformatics, speech recognition, and computer vision. Transformer has set a new standard for training large-scale models and producing high-quality outputs by working more effectively with data.

In conclusion, the Transformer architecture has revolutionized the world of artificial intelligence with both its theoretical innovations and practical achievements, and will continue to play a critical role in the design of future artificial intelligence systems. It is anticipated that the flexibility and power provided by this architecture will further advance artificial intelligence applications

metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**FIGURE-3**

This figure illustrates the process of **Self-Attention** and its role in the **Transformer architecture**, where input tokens are processed through multi-head attention and attention scores to generate output tokens.

**Conclusion**

As deep learning continues to shape advancements in various fields, understanding the key techniques that power these models is essential for driving innovation. Tokenization, embedding, pre-training, fine-tuning, self-attention, and transformer architecture are foundational concepts that enable models to perform tasks with high efficiency and accuracy.

Tokenization is a technique that simplifies data manipulation by breaking data into smaller, more manageable components. This process makes it easier to work with and analyze data. Embedding, on the other hand, serves as a representation of these pieces of data, or tokens, in a vector space. Embedding converts tokens into numerical vectors, allowing their relationships and meanings to be modeled and enabling more advanced data analysis and machine learning tasks. Tokenization and embedding are, together, the two fundamental processes in natural language processing and other data-driven processing.

Pre-training and fine-tuning are the basic methods that enable deep learning models to show high performance in both general and specific tasks. While pre-training enables the acquisition of generalized knowledge across large data sets, fine-tuning increases model accuracy by making this knowledge specific to the target task. When these two methods are used together, higher efficiency and accuracy can be achieved with less data. Research shows that particularly fine-tuned models achieve effective results on target tasks. As a result, pre-training and fine-tuning processes have become an indispensable tool for modern deep learning applications.

The self-attention mechanism has revolutionized deep learning by enabling models to capture intricate relationships within data sequences. Its ability to focus dynamically on relevant elements in a sequence has made it a cornerstone of modern machine learning architectures like Transformers. By providing a deeper understanding of how self-attention works and its computational steps, this report sheds light on why it is a key enabler of innovations in diverse domains. As research continues, self-attention is poised to further advance the capabilities of artificial intelligence.

Transformer architecture has opened the doors to a new era in sequential data processing and natural language processing. It has overcome the limitations of traditional models such as RNN and LSTM, and has provided a great advantage especially with its capacity to understand long-distance contexts and process data in parallel. Innovative approaches such as the self-attention mechanism and multiple attention heads have enabled the model to understand complex relationships and process different contexts simultaneously. Transformer, which is successfully used in many areas such as text translation, summarization, language understanding, and image processing, is not limited to NLP but can also be effectively applied to other types of data. This architecture forms the basis of modern artificial intelligence models today and directs developing technologies.

metin, ekran görüntüsü, çizgi, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

**FIGURE-4**

This figure illustrates the comprehensive workflow, outlining the key processes from **Data Preparation** to the **Transformer Architecture**, highlighting stages such as **Pre-Training**, **Fine-Tuning**, **Tokenization**, **Embedding**, and the role of **Self-Attention**.

**APPENDIX A**

ekran görüntüsü, metin, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Figure-5: Visualization of MFCC Features for Audio Data**

This figure visualizes the **MFCC (Mel Frequency Cepstral Coefficients)** features extracted from your **audio recording**over time, where color gradients represent the intensity and variation of different frequencies.

metin, ekran görüntüsü, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Figure-6: Token Changes Over Time for Audio Data**

This graph shows the changes in **MFCC coefficients** after performing **tokenization** on your **audio data**. It illustrates how the token values fluctuate over time and how different coefficients change with time.

metin, ekran görüntüsü, çizgi, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Figure -7: Embedding Feature Changes Over Time for Audio Data**

This graph shows the changes in the **embedding** features over time, derived from the **audio data**. It demonstrates how each embedding evolves and how different embeddings represent various characteristics of the audio data over time.

**References**

1. Sebastian Raschka. *Build LLMs*.
   * [Book Link (if applicable, please specify)]
2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention is All You Need*. Advances in Neural Information Processing Systems (NeurIPS), 30.
   * [Paper Link](https://arxiv.org/abs/1706.03762)
3. Aston Zhang, Zachary C. Lipton, Mu Li, & Alexander J. Smola. *Learning Transferable Visual Models From Natural Language Supervision*. *Dive into Deep Learning*.
   * [Book Link](https://d2l.ai/)
4. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). *Language Models are Few-Shot Learners*. Advances in Neural Information Processing Systems, 33.
   * [Paper Link](https://arxiv.org/abs/2005.14165)
5. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving Language Understanding by Generative Pre-Training*.
   * [Paper Link](https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf)
6. Wang, A., et al. (2021). *Measuring Information Transfer*.
   * [Paper Link](https://arxiv.org/abs/2106.04615)