
**Cairo University**

**Faculty of Engineering**

**Credit-Hour System Communication and Computer Engineering**

**CCEN 481: Graduation Project-2**

**Spring 2023**

Speak News

# Prepared by:

Ahmed Ehab Farghal Youssef Qadry Hashem

Mohamed Amr Afifi Abdallah Wael Marzouk

# Supervised by:

Dr. AbdElMoniem Bayoumi

**Abstract**

Traditional news reading methods often feel tedious and time-consuming, leading to decreased engagement and information overload. To address these challenges, we present an interactive news summarizer system that leverages avatar narration and text-to-speech conversion techniques. Our solution aims to revolutionize news consumption by providing concise news summaries that can be effortlessly listened to or visually experienced through avatars.

Recognizing the growing dissatisfaction with traditional news reading formats and the need for efficient information consumption, our system tackles these issues head-on. By utilizing advanced natural language processing and machine learning techniques, we automatically generate comprehensive news summaries, condensing key information from articles.

Our application offers users an enhanced news consumption experience by providing concise news summaries through avatar narration and text-to-speech conversion. Users can effortlessly listen to or visually experience news articles, overcoming the boredom and time constraints associated with traditional reading methods. By condensing key information using advanced natural language processing techniques, our system ensures accurate and digestible summaries. The integration of text-to-speech conversion generates immersive audio, while avatar narration offers a visually engaging alternative. Users will benefit from efficient and personalized news consumption, catering to their preferences and enabling them to stay informed conveniently.

**molakhas**

# Acknowledgement

Firstly, we would like to thank Allah for giving us power, tolerance, and persistence, to overcome all the hardships we faced along the way. We would also like to express our sincere gratitude to our advisor Dr. AbdElMoniem Bayoumi for the continuous support, unparalleled mentorship, and guidance. His guidance helped us in all the time of research and writing of this thesis. Last but not least, we would like to thank our families for their patience, encouragement, and continuous support they have given us along the way.

**Table of Contents**

[Chapter 1 8](#_Toc136742913)

[Introduction 8](#_Toc136742914)

[1.1 Project Idea 9](#_Toc136742915)

[1.2 Motivation 10](#_Toc136742916)

[1.3 Document overview 12](#_Toc136742917)

[Chapter 2 15](#_Toc136742918)

[Necessary Background 15](#_Toc136742919)

[2.1 Basics for summarization model 16](#_Toc136742920)

[2.1.1 Positional Embedding 16](#_Toc136742921)

[2.1.1.1 Sinusoidal Positional Embeddings 16](#_Toc136742922)

[2.1.1.2 Learned Positional Embeddings 17](#_Toc136742923)

[2.1.2 Attention Mechanism 17](#_Toc136742924)

[2.1.2.1 Self-Attention 18](#_Toc136742925)

[2.1.2.2 Masked Attention 19](#_Toc136742926)

[2.1.2.3 Cross-Attention 20](#_Toc136742927)

[2.1.3 BERT 21](#_Toc136742928)

[2.1.4 Beam Search Decoding 23](#_Toc136742929)

[2.2 Basics for text-to-emotion model 24](#_Toc136742930)

[2.2.1 TF -IDF 24](#_Toc136742931)

[2.2.2 LSTM 25](#_Toc136742932)

[2.2.3 Word Embedding 27](#_Toc136742933)

[2.2.3 GloVe (Global Vectors) 28](#_Toc136742934)

[2.3 Basics for Text to speech 29](#_Toc136742935)

[2.3.1 Signal Windowing 29](#_Toc136742936)

[2.3.2 Short-Time Fourier Transform (STFT) 31](#_Toc136742937)

[2.3.3 Mel-Spectrogram 33](#_Toc136742938)

[2.3.4 Mel Scale 34](#_Toc136742939)

[2.3.5 Griffin Lim Algorithm 34](#_Toc136742940)

[Chapter 3 36](#_Toc136742941)

[Literature Review 36](#_Toc136742942)

[3.1 Models Approaches 37](#_Toc136742943)

[3.2 Evaluation Metrics: 39](#_Toc136742944)

[3.3 Real-World Application 40](#_Toc136742945)

[Chapter 4 System Architecture 42](#_Toc136742946)

[Chapter 5 44](#_Toc136742947)

[Models 44](#_Toc136742948)

[5.1 Summarizer 45](#_Toc136742949)

[5.2 Text-to-Emotion 46](#_Toc136742950)

[5.2.1 Input 46](#_Toc136742951)

[5.2.2 Input dataset 46](#_Toc136742952)

[5.2.3 Pre-processing 46](#_Toc136742953)

[5.2.4 Tokenization and word embedding 47](#_Toc136742954)

[5.2.4.1 Tokenization 47](#_Toc136742955)

[5.2.4.2 Word Embedding 48](#_Toc136742956)

[5.2.5 Full Model Architecture 50](#_Toc136742957)

[5.2.6 Using LSTM Over RNN 50](#_Toc136742958)

[5.2.7 Model Evaluation 52](#_Toc136742959)

[5.3 News Genre Classifier 53](#_Toc136742960)

[5.3.1 Input 53](#_Toc136742961)

[5.3.2 Pre-processing 53](#_Toc136742962)

[5.3.3 Model Architecture 54](#_Toc136742963)

[5.3.4 Model Evaluation 54](#_Toc136742964)

[5.3 Text to Speech 55](#_Toc136742965)

[5.3.1 Input 55](#_Toc136742966)

[5.3.2 Input Dataset 55](#_Toc136742967)

[5.3.3 Pre-processing 55](#_Toc136742968)

[5.2.5 Full Model architecture 57](#_Toc136742969)

**List of Figures**

[Figure 1 Decline in printed news paper 10](file:///C:\Users\abdal\gp\our_book\GP2%20Report.docx#_Toc136741529)

[Figure 2 Median age for news consumption 11](file:///C:\Users\abdal\gp\our_book\GP2%20Report.docx#_Toc136741530)

[Figure 3 Sinusoidal Positional Encoding Rule Formula 16](#_Toc136741531)

[Figure 4 The 128-dimensional positonal encoding for a sentence with the maximum lenght of 50. Each row represents the embedding vector 16](#_Toc136741532)

[Figure 5 Self Attention Mechanism 17](file:///C:\Users\abdal\gp\our_book\GP2%20Report.docx#_Toc136741533)

[Figure 6 Masked Self-attention Mechanism 19](#_Toc136741534)

[Figure 7 Cross attention Mechanism 20](file:///C:\Users\abdal\gp\our_book\GP2%20Report.docx#_Toc136741535)

[Figure 8 Overall pre-training and fine-tuning procedures for BERT 22](#_Toc136741536)

[Figure 9 Example of Beam Search with k = 2 23](#_Toc136741537)

[Figure 3 Flow of Hidden State through RNN. 25](file:///C:\Users\abdal\gp\our_book\GP2%20Report.docx#_Toc136741538)

[Figure 4 LSTM Cell 25](file:///C:\Users\abdal\gp\our_book\GP2%20Report.docx#_Toc136741539)

[Figure 5 GloVe Word Embedding 51](file:///C:\Users\abdal\gp\our_book\GP2%20Report.docx#_Toc136741540)

[Figure 6 Text-to-Emotion Model Architecture 52](file:///C:\Users\abdal\gp\our_book\GP2%20Report.docx#_Toc136741541)

**List of Tables**

# List of Abbreviations

|  |  |
| --- | --- |
| TF | Term Frequency |
| IDF | Inverse Document Frequency |
| TF-IDF | Term Frequency-Inverse Document frequency |
| STFT | Short-time Fourier transform |
| RNN | Recurrent Neural Network |
| LSTM | Long Short-Term Memory |
| GloVe | Global Vectors for Word Representation |
| UNK | Unknown |
| SVM | Support Vector Machines |
| Seq2Seq | Sequence to Sequence |

# Contacts

# Team Members

|  |  |  |
| --- | --- | --- |
| Name | Email | Phone number |
| Ahmed Ehab Farghal (A7la team leader) | [ahmedehabb17@gmail.com](mailto:ahmedehabb17@gmail.com) | 01008536356 |
| Youssef Qadry Hashem | [yousefqadryy@gmail.com](mailto:yousefqadryy@gmail.com) | 01117800943 |
| Mohamed Amr Afifi | [mohamed.amr.afifi@gmail.com](mailto:mohamed.amr.afifi@gmail.com) | 01066991087 |
| Abdallah Wael Marzouk | [abdallahwm1@gmail.com](mailto:abdallahwm1@gmail.com) | 01030702155 |

# Supervisor

|  |  |
| --- | --- |
| Supervisor name | Email |
| AbdElMoniem Bayoumi | abdelmoniem.bayoumi@gmail.com |

# Chapter 1

# Introduction

Introduction

## Project Idea

The project aims to develop an innovative and immersive news summarizer system that revolutionizes the way users consume news articles. By incorporating avatar narration, text-to-speech conversion, emotion synthesis, and genre classification, the system offers a comprehensive and engaging news consumption experience.

The avatar narration component adds a visually captivating element to the news summaries. Users can select from a diverse range of avatars, each with its unique style and appearance, to narrate the news content. This visually immersive feature creates a dynamic and interactive experience, enhancing user engagement and making the news consumption process more enjoyable.

The text-to-speech conversion feature employs advanced technologies such as neural networks and deep learning models to synthesize natural and high-quality speech. This enables users to listen to the news summaries in an immersive and convenient audio format. The system provides an engaging and accessible alternative for individuals who prefer auditory information consumption or have visual impairments.

To further enhance the emotional impact of the news summaries, the system incorporates a text-to-emotion model. This model analyzes the sentiment and emotional tone of the news content and synthesizes corresponding emotions. Users have the option to experience the news summaries with added emotional cues, such as joy, sadness. This feature aims to create a more immersive and impactful news consumption experience by evoking emotional responses and increasing user engagement.

Additionally, the system includes a news genre classifier that categorizes news articles into various genres or topics. By employing machine learning algorithms, the classifier identifies the genre of each news article, such as politics, sports, technology, or entertainment. Users can explore specific topics of interest or access news summaries from their preferred genres, allowing for a personalized and focused news consumption experience.

The project involves data collection, preprocessing, training and integrating the text-to-emotion model and genre classifier and developing an intuitive user interface. To evaluate the system's performance, quantitative metrics such as accuracy of emotion synthesis and genre classification will be measured. User feedback will be collected through surveys and usability studies to assess the system's effectiveness in providing an immersive and satisfying news consumption experience.

By integrating avatar narration, text-to-speech conversion, emotion synthesis, and genre classification, our project aims to offer users a comprehensive and immersive news summarizer system. This innovative approach enhances engagement, personalization, and emotional impact, providing a more enjoyable and tailored news consumption experience. Whether users prefer visual, auditory, or emotionally engaging news consumption, our system caters to diverse preferences, making news consumption a dynamic and fulfilling activity.

To evaluate our system's effectiveness, we conducted comprehensive experiments using diverse news datasets. We compared the generated summaries against human-generated summaries and assessed metrics such as ROUGE scores. User studies were also conducted to gauge usability, comprehension, and overall satisfaction with the text-to-speech conversion and avatar narration features.

Results demonstrate the system's ability to provide accurate and concise news summaries. Users found the combination of avatar narration and text-to-speech conversion to be engaging, accessible, and greatly improved their news consumption experience. Our interactive news summarizer holds significant potential for enhancing news accessibility and facilitating efficient information consumption, accommodating a wide range of user preferences and needs.

## Motivation and Justification

The motivation behind our project stems from the growing need to address the challenges and limitations of traditional news consumption methods. Reading lengthy news articles can be time-consuming, overwhelming, and often leads to information overload. Furthermore, with the increasingly fast-paced nature of modern life, many individuals find it difficult to dedicate sufficient time to read through extensive news content. This results in a decreased engagement with news and a potential lack of awareness on important current events.

Moreover, we recognize that different individuals have varying preferences when it comes to consuming information. While some may prefer reading, others may find it more convenient and enjoyable to listen to news summaries or experience them in a visually engaging manner. By catering to diverse preferences and providing alternative modes of news consumption, we aim to make news more accessible and engaging for a broader range of individuals.

Additionally, our motivation stems from the desire to enhance the emotional connection that users have with news content. Traditional news articles often fail to evoke the emotional impact that real-life events may warrant. By incorporating emotion synthesis into our system, we aim to bring an added layer of engagement and impact to the news summaries, enabling users to connect on a deeper level with the stories they encounter.

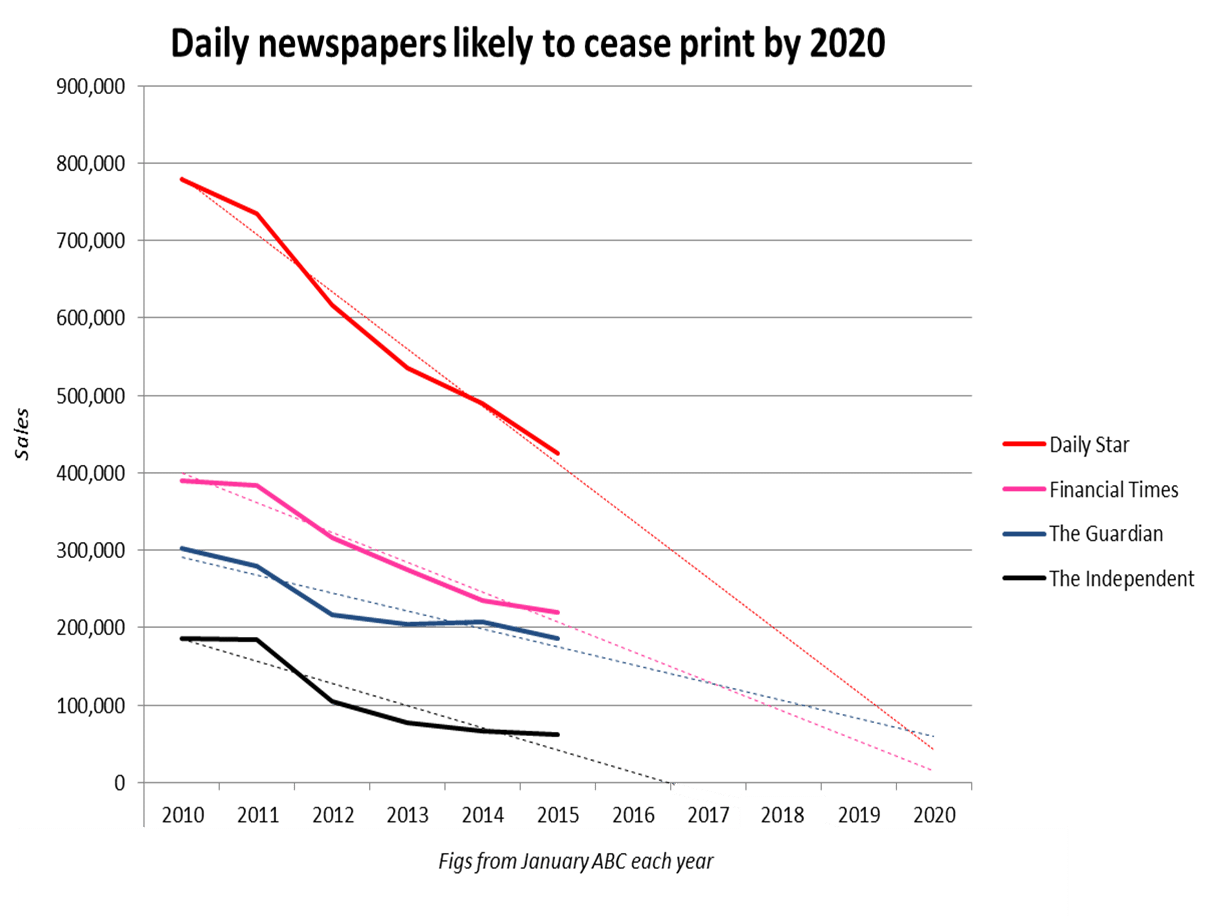
We believe that this decline in printed newspapers underscores the need for innovative solutions that cater to the changing habits and preferences of news consumers. As people increasingly rely on digital platforms for accessing news, there is a growing demand for convenient, engaging, and personalized ways of consuming news content.

Figure 1 Decline in printed news paper

Our project is further motivated by the observation that the majority of newspaper readership now consists of older individuals, while younger generations are increasingly turning to online platforms for news consumption. This generational shift in news consumption habits highlights the need to adapt traditional news formats to cater to the preferences of the younger, tech-savvy demographic.

With our interactive news summarizer system, we aim to bridge the gap between generations by offering a modern and technologically advanced solution that caters to the preferences of both older and younger readers. By providing visually immersive avatars, text-to-speech conversion, emotion synthesis, and genre classification, we create an interactive and dynamic news consumption experience that appeals to individuals of all age groups.

We believe that by delivering news in a more interactive and engaging way, we can capture the interest of younger generations who are accustomed to digital media and prefer more interactive and personalized content experiences. By providing a seamless transition from traditional newspapers to an interactive digital platform, we can attract a broader audience and encourage younger individuals to engage with news content more actively.

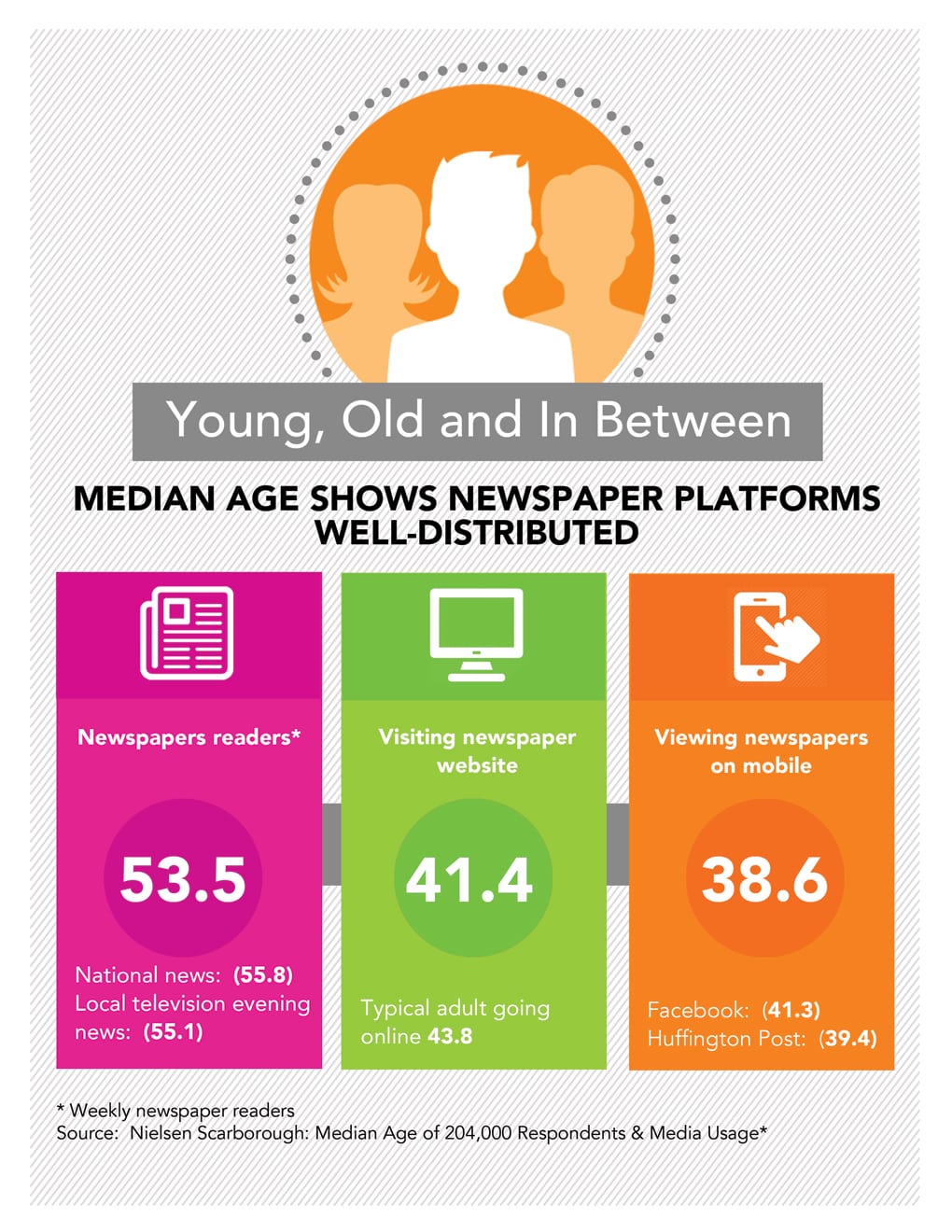


Figure 2 Median age for news consumption

## Document Organization

This document provides an overview of the structure and content of our research paper on creating a news summarizer with avatar narration and text-to-speech conversion. The paper is organized into several chapters that explore different aspects of the project and its implementation.

*Chapter 1: Introduction*

This chapter introduces the background and motivation behind the project, highlighting the decline in traditional newspaper readership and the need for innovative news consumption methods. It also presents the objectives, scope, and significance of the research.

*Chapter 2: Necessary Background*

In this chapter, we delve into the fundamental concepts and technologies that form the foundation of our news summarizer system. We discuss concepts such as word embedding, GloVe vectors, face encoders, Generative Adversarial Networks (GANs), Long Short-Term Memory (LSTM) models, and mel spectrograms. This chapter provides a comprehensive understanding of the underlying technologies employed in our system.

*Chapter 3: Literature Review*

The literature review chapter explores existing research and developments related to news summarization, text-to-speech conversion, emotion synthesis, and genre classification. We analyze various studies, methodologies, and approaches to gain insights into the current state of the field and identify gaps that our research aims to address.

*Chapter 4: System Architecture*

In this chapter, we present the overall architecture and design of our full project system. We discuss the components, their interactions, and the flow of data within the system. This chapter provides a comprehensive overview of how the different modules work together to deliver an immersive and interactive news consumption experience.

*Chapter 5: Model Implementation and Evaluation*

Chapter 5 focuses on the implementation details of each model employed in the system. We provide a detailed description of the implementation steps, model architectures, and training processes. Additionally, we discuss the evaluation methodologies used to assess the performance and effectiveness of the models.

*Chapter 6: Tools Used*

In this chapter, we outline the tools, frameworks, and libraries utilized in the development of our news summarizer system. We provide an overview of the technologies employed for data collection, preprocessing, training models, and building the user interface. This chapter serves as a reference for researchers and developers interested in replicating or extending our work.

*Chapter 7: Experiments*

TODO RNNs fel text emotion

model mashtaghalsh fel summarizer

Garabna ne3mel sync net loss bs fashalna

*Chapter 8: Future Work*

The final chapter highlights potential avenues for future research and enhancements to our news summarizer system. We discuss possible improvements to the existing models, exploration of additional features, and potential collaborations with other technologies or platforms. This chapter provides a roadmap for further advancements in the field of news summarization and interactive news consumption.

# Chapter 2

# Visibility Study

## Target Customers

# Chapter 3

# Literature Survey

## 3.1 Background on Summarization

## (todo) tokenization explanation

### 3.1.1 Non-Engineering Background

Summarization is a crucial task in many NLP applications, such as document summarization, news aggregation, chatbots, and content curation. It helps in efficiently processing and understanding large volumes of textual data, enabling users to quickly grasp the main ideas and relevant information without having to go through the entire text.

#### 3.1.1.1 Summarization Types

There are two main types of summarization techniques: extractive summarization and abstractive summarization.

1. **Extractive Summarization**: Extractive summarization involves selecting and extracting the most important sentences or phrases from the original text to create a summary. These sentences are typically chosen based on their relevance, informativeness, and coherence. In extractive summarization, the summary consists of actual sentences or phrases that are present in the original text. This approach is akin to human summarization, where key information is manually identified and extracted.
2. **Abstractive Summarization**: Abstractive summarization, on the other hand, involves generating a summary that may contain words, phrases, or even sentences that are not explicitly present in the original text. Instead of extracting sentences directly, abstractive summarization models generate new phrases and sentences that capture the essence of the original text. This approach requires a deeper understanding of the text and the ability to generate coherent and contextually relevant language.

Both extractive and abstractive summarization tasks have their advantages and challenges. Extractive summarization is relatively simpler to implement, as it involves selecting sentences directly from the text. It can preserve the exact wording and factual information from the original text. Abstractive summarization, on the other hand, allows for more flexibility and creativity in generating summaries. It can capture the main ideas and provide concise summaries even when the original text is extensive. However, abstractive summarization is often more challenging as it requires a deeper understanding of the text and the ability to generate coherent and contextually appropriate language.

Here's a comparison of the output for extractive and abstractive summarization using the same original text:

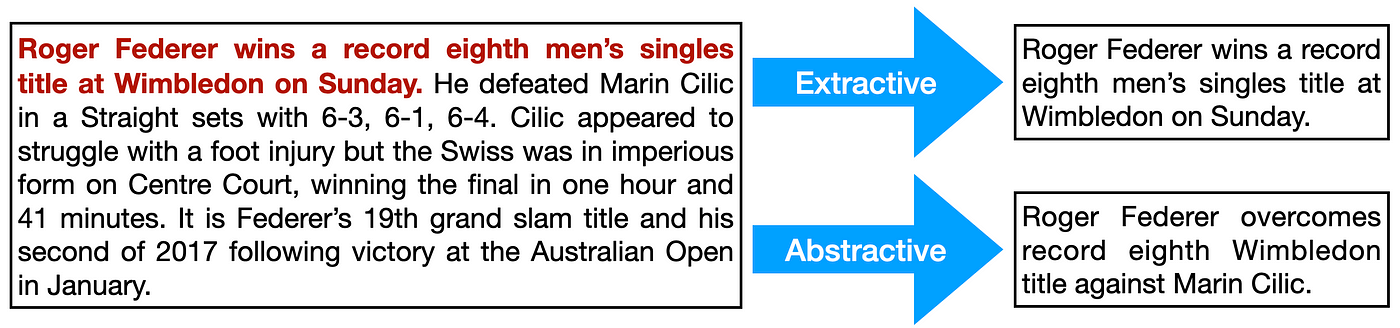


Figure 3 Extractive vs Abstractive Summarization Task

Note that in the extractive summary, the sentences are directly extracted from the original text without any rephrasing or modification. On the other hand, the abstractive summary includes a more concise and paraphrased version of the original information, presenting it in a way that captures the key points while using different wording.

### 3.1.2 Engineering Background

#### 3.1.2.1 Engineering Background for Abstractive Summarizer

Sequence-to-Sequence Models: Abstractive summarization involves generating a summary that may contain new phrases or sentences not present in the source text. Sequence-to-sequence (Seq2Seq) models, based on recurrent neural networks (RNNs) or transformers, are commonly used for abstractive summarization. These models learn to map the source text to a target summary by training on large amounts of paired data.

###### 3.1.2.1.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have been widely used in abstractive summarization tasks due to their ability to model sequential data. RNNs are a class of neural networks that have connections between the hidden units, allowing them to maintain an internal memory and process sequences of inputs.

In the context of abstractive summarization, RNNs can be used to generate summaries by sequentially predicting each word or token in the output sequence. The key idea is to use the hidden state of the RNN to capture the context and generate the next word based on the previous words generated. This allows the model to consider the context and generate more fluent and coherent summaries.

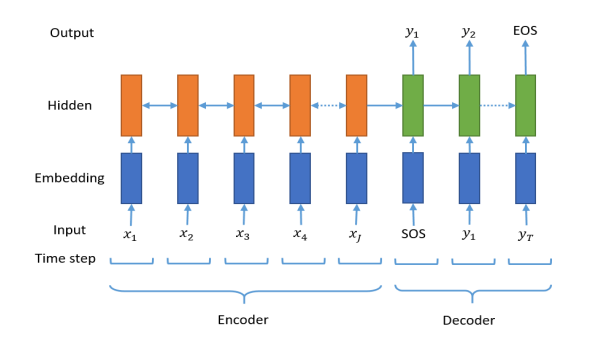


Figure 4 RNN for Abstractive Summarization

###### 3.1.2.1.2 Encoder Decoder Transformers

Encoder-decoder transformer models have emerged as a powerful approach for abstractive summarization tasks, offering several advantages over recurrent neural networks (RNNs). These models leverage the transformer architecture, which relies on self-attention mechanisms to capture long-range dependencies and process input sequences in parallel.

###### 3.1.2.1.2.1 Encoder Decoder Transformers vs RNN

Here are some reasons why encoder-decoder transformer models are often considered superior to RNNs in abstractive summarization:

1. **Capturing long-range dependencies**: Transformers are designed to capture long-range dependencies more effectively compared to RNNs. The self-attention mechanism allows each word to attend to all other words in the sequence, enabling the model to capture global context and dependencies without the limitations of sequential processing.
2. **Parallel computation**: Transformers can process the input sequence in parallel, as opposed to the sequential nature of RNNs. This parallelism significantly speeds up training and inference, making transformer models more efficient.
3. **Reduced vanishing gradient problem**: Transformers mitigate the vanishing gradient problem encountered in RNNs. The self-attention mechanism helps the model to propagate gradients more effectively, allowing for better gradient flow during training and more stable optimization.
4. **Better modeling of global context**: The transformer architecture, with its self-attention mechanism, enables the model to consider the entire input sequence simultaneously. This ability to capture global context is crucial for abstractive summarization, as it allows the model to understand the overall theme, important information, and structural relationships in the source text.
5. **Improved generation of diverse and coherent summaries**: Encoder-decoder transformers tend to produce more diverse and coherent summaries compared to RNN-based approaches. The models can leverage the attention mechanism to focus on different parts of the input sequence while generating each word, leading to more creative and varied summaries.

Overall, encoder-decoder transformer models have demonstrated superior performance in abstractive summarization tasks due to their ability to capture long-range dependencies, model global context effectively, and generate diverse and coherent summaries.

###### 3.1.2.1.2.2 Encoder Decoder Transformers Architecture

In the context of abstractive summarization, encoder-decoder transformers consist of two main components:

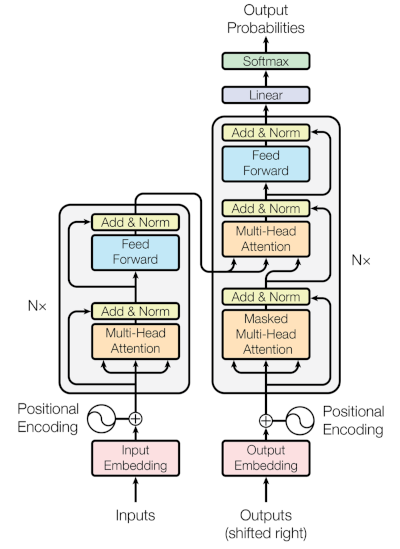
1. Encoder: The encoder takes the input source text and encodes it into a set of high-dimensional representations, often referred to as embeddings. Each word or token in the input sequence is transformed into a dense vector representation, capturing its contextual information and relationship with other words in the sequence. The self-attention mechanism in the transformer architecture allows the encoder to consider the entire input sequence simultaneously, effectively capturing global dependencies.
2.  Decoder: The decoder takes the encoded representations from the encoder and generates the output summary. It predicts each word or token in the summary by attending to the encoded representations and considering the previously generated words. The decoder also utilizes self-attention mechanisms to capture the dependencies between the generated words and the input sequence.

Figure 5 Encoder Decoder Transformer

3.1.2.1.2.2.1 Encoder Architecture

The Encoder Transformers consist of the following components:

3.1.2.1.2.2.1.1 Input Embedding

The input embedding is a crucial component in the Encoder-Decoder Transformer model. It is responsible for converting input tokens into continuous representations, commonly known as word embeddings. Word embeddings are dense vector representations that capture the semantic and syntactic properties of words.

(TODO) reference afifis word embedding may make them common.

3.1.2.1.2.2.1.2 Positional Encoding

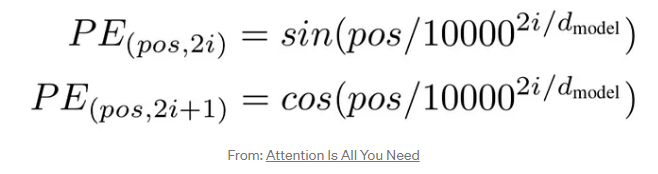
In Transformers, positional embeddings are used to provide the model with information about the position of each word in the input sequence. They enable the model to capture the sequential order of the words, which is crucial for understanding the context and dependencies within the sequence. There are two common types of positional embeddings used in Transformers: sinusoidal and learned positional embeddings.

Both types of positional embeddings serve the purpose of providing the model with positional information. The choice between sinusoidal and learned positional embeddings depends on the specific task and the availability of training data. Sinusoidal embeddings are commonly used when the model has limited training data or when simplicity is preferred. Learned embeddings, on the other hand, are used when there is sufficient training data available, and the model can benefit from capturing more intricate positional dependencies.

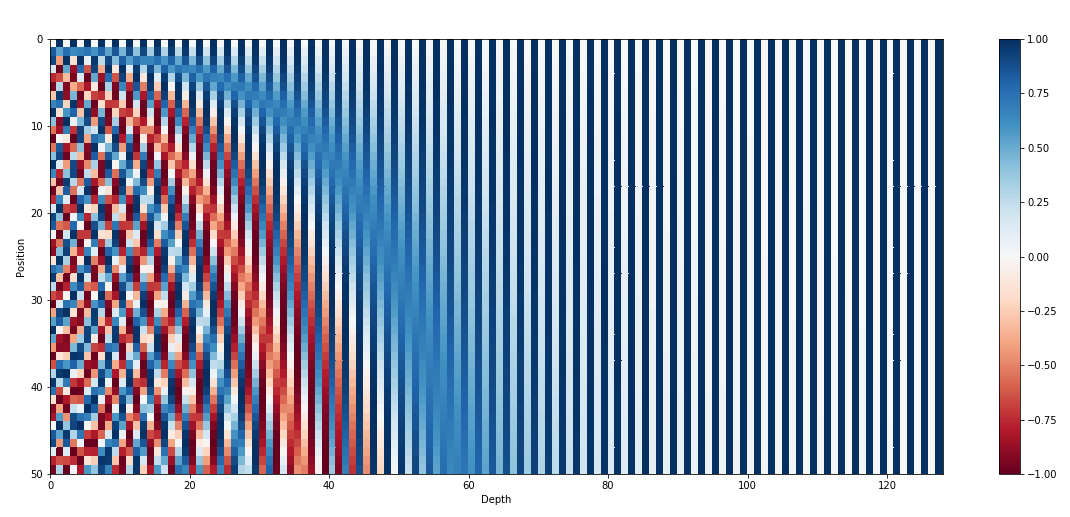
Overall, positional embeddings are an essential component of Transformers, enabling the model to understand the sequential order of words and capture contextual dependencies within the input sequence. They play a crucial role in allowing the model to effectively process and generate coherent and contextually appropriate outputs.

3.1.2.1.2.2.1.2.1 Sinusoidal Positional Embeddings

Sinusoidal positional embeddings are a fixed set of embeddings that are added to the input word embeddings to convey positional information. These embeddings are based on sine and cosine functions of different frequencies and amplitudes. The position of each word in the sequence corresponds to a unique combination of these sinusoidal functions. Sinusoidal positional embeddings have the advantage of being deterministic and easily computed, as they do not require any additional parameters to learn. However, they do not capture any specific patterns or dependencies in the data.



*Figure 6 Sinusoidal Positional Encoding Rule Formula*



*Figure 7 The 128-dimensional positonal encoding for a sentence with the maximum lenght of 50. Each row represents the embedding vector*

3.1.2.1.2.2.1.2.1 Learned Positional Embeddings

Learned positional embeddings, on the other hand, are trainable parameters that the model learns during the training process. Instead of using fixed sinusoidal functions, the model dynamically learns the positional embeddings based on the input sequence. Learned positional embeddings have the advantage of capturing more complex patterns and dependencies in the data. The model can adapt the embeddings to better represent the relationships between positions in the sequence. However, they come with the additional parameter overhead and require more computational resources during training.

3.1.2.1.2.2.1.3 Self-Attention

In self-attention, the model attends to different positions within the input sequence to capture relationships and dependencies between words. The self-attention mechanism involves the following steps:

* Input Representation: Each word in the input sequence is transformed into Query (Q), Key (K), and Value (V) vectors by applying learned linear transformations to the word embeddings.
* Scoring Function: The self-attention mechanism calculates the similarity between each Query vector and all Key vectors in the input sequence. The dot-product or scaled dot-product scoring functions are commonly used.
* Attention Weights: The similarity scores obtained from the scoring function are passed through a softmax function to obtain attention weights. These weights represent the importance or relevance of each word in the input sequence.
* Weighted Sum: The attention weights are applied to the Value vectors using element-wise multiplication, resulting in a weighted sum. This step allows the model to focus on different parts of the input sequence based on their relevance to the current position.
* Multi-Head Self-Attention: Transformers often employ multiple self-attention heads to capture different dependencies and aspects within the input sequence. Each attention head performs the same process described above but with different learned weight matrices.

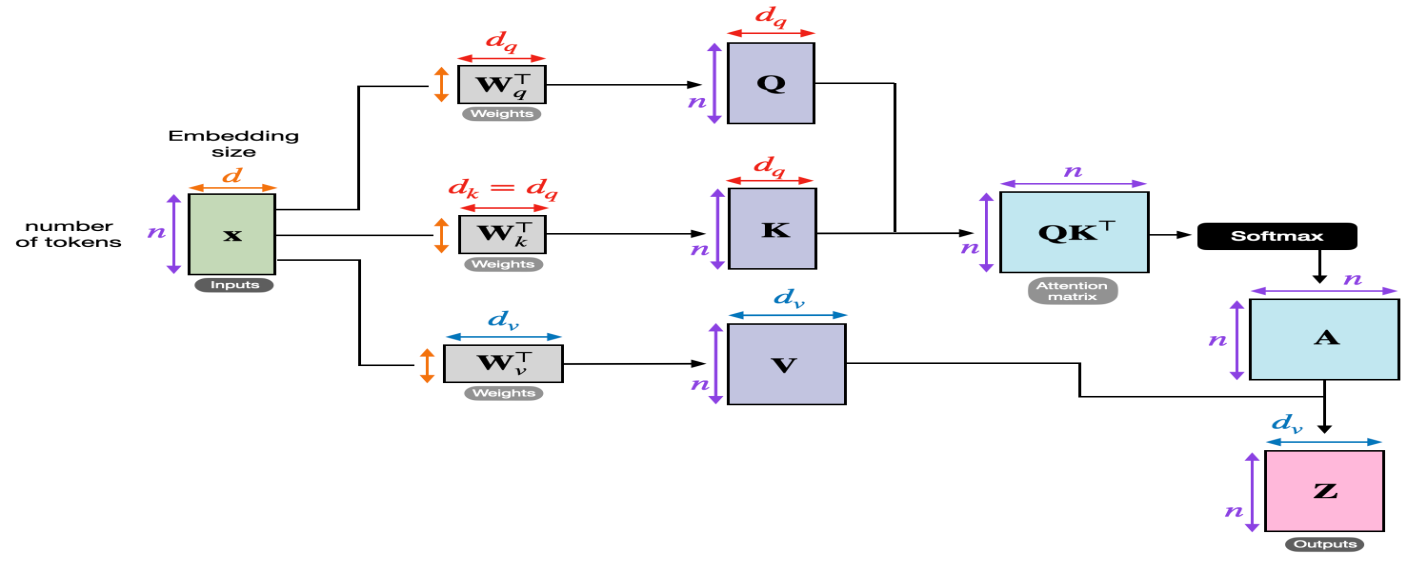


Figure 8 Self Attention Mechanism

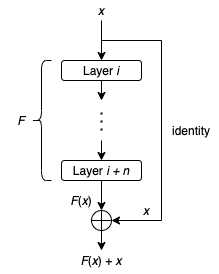
3.1.2.1.2.2.1.3 Residual Connections

Figure 9 Residual Connections

Residual connections, also known as skip connections, are an essential component in the Transformer architecture. They were introduced to address the vanishing gradient problem that can occur in deep neural networks.

In the Transformer, residual connections are used to connect the output of one layer to the input of a subsequent layer. This is achieved by adding the input of a layer to its output, element-wise. The purpose of this connection is to pass the original information from the input directly to the deeper layers of the network.

3.1.2.1.2.2.1.3 Layer Normalization

Layer normalization is a technique used in deep learning models, including Transformer architectures, to normalize the activations within a layer. It aims to address the issue of internal covariate shift, where the distribution of inputs to each layer changes during training, making it harder for the model to learn effectively.

The normalization process involves subtracting the mean and dividing by the standard deviation. This centers the activations around zero and scales them to have a unit variance. After normalization, the activations are transformed using learned parameters, such as a scale and shift, to allow the model to adapt and learn the optimal representation.

Layer normalization offers several advantages. Firstly, it helps with gradient propagation by reducing the internal covariate shift and stabilizing the training process. It can lead to faster convergence and improved training dynamics.

Secondly, layer normalization is less dependent on the batch size compared to batch normalization. This makes it suitable for scenarios where the batch size may vary or when batch normalization is not applicable, such as during inference on a single example.

Lastly, layer normalization has been observed to have a regularizing effect, similar to batch normalization. It can help prevent overfitting by introducing a form of noise to the activations and reducing the sensitivity to the magnitude of inputs.

(TODO) see post norm

3.1.2.1.2.2.1.3 Position-wise Feed-Forward Neural Network

A Feed-Forward Neural Network (FNN) is a type of artificial neural network where the information flows in one direction, from the input layer through one or more hidden layers to the output layer. It is called "feed-forward" because the connections between the neurons do not form cycles or loops.

The FNN is composed of multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to all neurons in the adjacent layers. The neurons in the hidden layers apply non-linear transformations to the input data, allowing the network to learn complex patterns and relationships in the data.

E:\8th Semester CCE\reports\ffn equation.PNG

Figure 10 Feed Forward Network

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1.

The main purpose of the feed-forward neural network is to map input data to the corresponding output. During the forward pass, the input data is fed into the network, and the activations of each neuron are computed based on the weighted sum of the inputs and the activation function applied to the sum. The activations are then passed to the next layer until the output layer is reached, which produces the final output.

Feed-forward neural networks are known for their ability to learn complex patterns and solve a wide range of tasks, including classification, regression, and pattern recognition. They have been successfully applied in various domains, including image recognition, natural language processing, and speech recognition.

3.1.2.1.2.2.2 Decoder Architecture

The Decoder Transformers consist of the following concepts:

3.1.2.1.2.2.2.1 Masked Multi-head Attention

Masked self-attention is a variant of the self-attention mechanism used in Transformers that includes a masking step to prevent positions from attending to future positions. It is particularly useful in scenarios where the output sequence is generated incrementally, such as in language modeling or autoregressive tasks. The cross-attention mechanism differs from self-attention in the following:

Masking: Before applying the attention weights, a mask is applied to the attention matrix to prevent positions from attending to future positions. The mask sets the attention weights for future positions to a very large negative value or zero, effectively making their contributions negligible during the weighted sum step.

By incorporating masking into the self-attention mechanism, masked self-attention ensures that each position can only attend to previous positions, preserving the autoregressive property of the generation process. This is particularly important when generating sequences incrementally, as it prevents the model from relying on future information during the generation of each position.

Masked self-attention is commonly used in language modeling tasks, where the model predicts the next word in a sequence given the previous context. By attending only to the past positions, the model can effectively capture dependencies and generate coherent and contextually appropriate sequences.

In summary, masked self-attention in Transformers restricts attention to previous positions by applying masking, allowing the model to generate sequences incrementally while maintaining the autoregressive nature of the generation process.

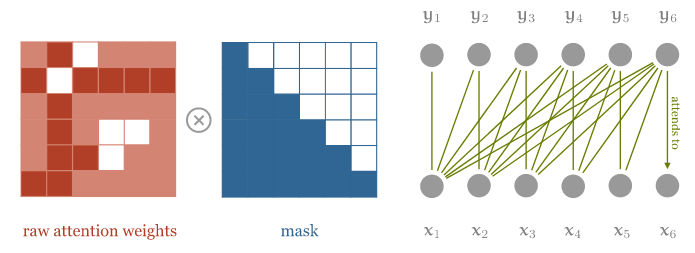


Figure 11 Masked Attention Mechanism

3.1.2.1.2.2.2.2 Cross Attention

Cross-attention allows the model to attend to different parts of the source sequence while generating the output sequence. It incorporates information from the source sequence to enhance the decoding process. The cross-attention mechanism differs from self-attention in the following:

* + Source and Target Representations: The encoder processes the source sequence and generates a series of encoded representations for each word. The decoder processes the target sequence and produces encoded representations for each word.
  + Query, Key, and Value: The decoder's hidden state at a particular time step is transformed into Query vectors (Q), while the encoder's hidden states are transformed into Key (K) and Value (V) vectors. These vectors allow the model to attend to different parts of the source sequence.
  + Scoring Function: The cross-attention mechanism calculates the similarity between the Query vectors from the decoder and the Key vectors from the encoder. The dot-product or scaled dot-product functions are commonly used for scoring.
  + Attention Weights: The similarity scores are passed through a soft-max function to obtain attention weights. These weights determine the importance of different parts of the source sequence during the generation of the target sequence.
  + Weighted Sum: The attention weights are applied to the encoder's Value vectors using element-wise multiplication, resulting in a weighted sum. This step combines the information from different parts of the source sequence based on their relevance to the current output.

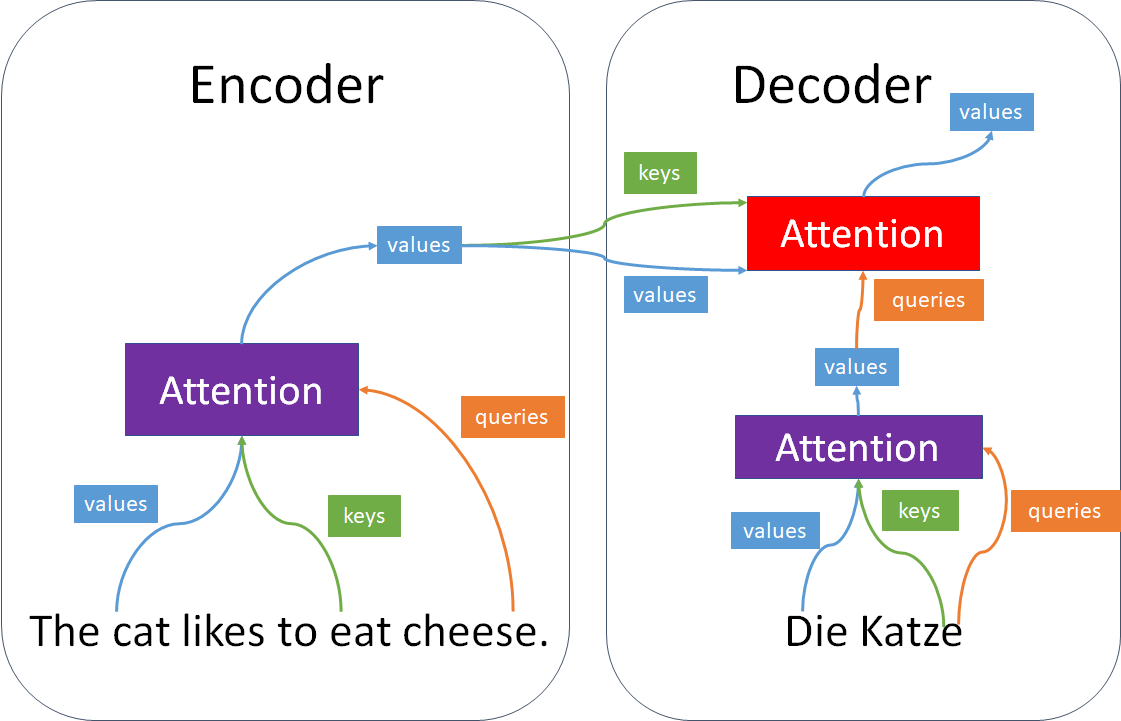


Figure 12 Cross attention Mechanism

3.1.2.1.2.2.3 After The Decoder

After the Decoder in the Transformer model processes the input sequence and generates hidden representations, the Linear and Softmax layers are commonly applied to produce the final output probabilities for each token in the output vocabulary.

The Linear layer is a fully connected layer that performs a linear transformation on the hidden representations. It projects the high-dimensional hidden representations into a space that matches the size of the output vocabulary. This linear transformation allows the model to learn different weights for each token in the output vocabulary, capturing the relationships between the hidden representations and the probability distribution over the tokens.

The Softmax layer is applied after the Linear layer to convert the transformed representations into a probability distribution. The softmax function takes the transformed values and normalizes them across the vocabulary, ensuring that the resulting values sum up to 1. This normalization process generates probabilities for each token, indicating the likelihood of each token being the next token in the generated output sequence.

The purpose of the Linear and Softmax layers is to generate meaningful probabilities over the output vocabulary, which allows the model to make informed decisions about the next token to generate based on the context and information encoded in the hidden representations. By applying these layers, the model can generate coherent and contextually relevant output sequences that are appropriate for the given task, such as language translation or text generation.

Together, the Linear and Softmax layers in the Decoder contribute to the generation of the final output probabilities, enabling the model to generate accurate and fluent sequences based on the learned representations and context.

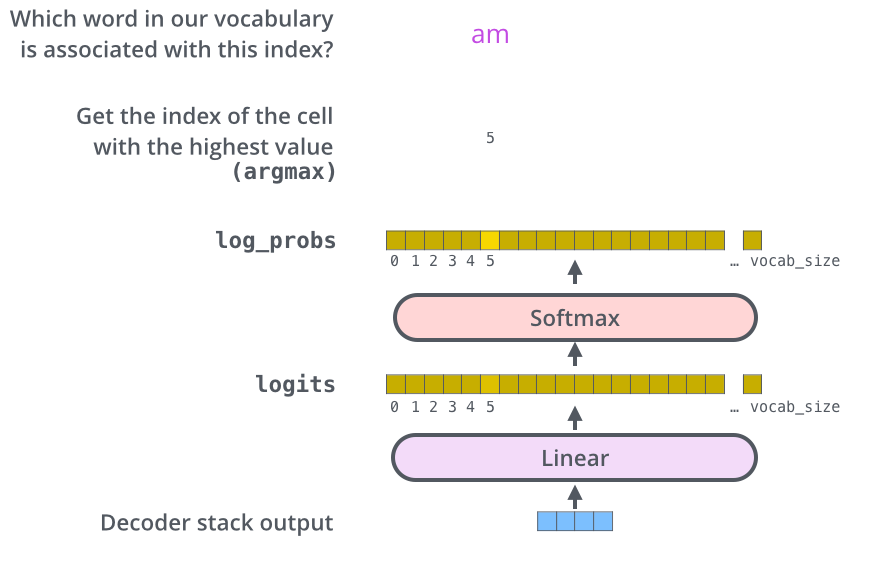


Figure 13 The Linear and Softmax Layers at the end

#### 3.1.2.2 Engineering Background for Extractive Summarizer

In an extractive summarization approach, the goal is to select and combine important sentences or phrases from the source document to create a summary. One way to achieve this is by using an encoder and a classifier.

The encoder in an extractive summarizer is same as the encoder explained in Abstractive Summarizer. It is responsible for encoding the source document into a fixed-dimensional representation. It can be a pretrained language model such as BERT or GPT, which is designed to capture the contextual information and semantic meaning of the input text. The encoder processes the input document and produces high-dimensional representations for each word or subword in the document.

Once the source document is encoded, the classifier comes into play. The classifier is a binary classifier that predicts whether a particular sentence or phrase should be included in the summary or not. It takes as input the encoded representations of the sentences or phrases in the document and applies a classification layer on top. The classification layer can be a simple logistic regression or a neural network-based classifier.

During the training phase, the classifier is trained using labeled data, where each sentence or phrase in the source document is annotated with a binary label indicating whether it should be included in the summary or not. The classifier learns to predict the correct labels based on the encoded representations of the input sentences or phrases.

(TODO) explain how you label data using rouge

In the summarization process, the classifier is applied to each sentence or phrase in the source document, and it predicts the probability of including that sentence or phrase in the summary. The sentences or phrases with the highest probabilities are selected and concatenated to form the final summary.

The extractive summarizer based on the encoder and classifier approach has the advantage of directly selecting and using sentences or phrases from the source document, which ensures that the summary maintains the original context and wording. However, it may struggle with generating coherent and fluent summaries as it relies on pre-existing sentences or phrases rather than generating new content.

Overall, the extractive summarization approach using an encoder and classifier is an effective method for selecting important content from the source document to create a concise summary.



Figure 14 Encoder and Classifier on top

Todo *OOV*

#### 3.1.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a powerful pre-trained language model that has revolutionized various natural language processing (NLP) tasks. It introduced the concept of bidirectional training, allowing the model to capture contextual information from both left and right contexts of a given word. Let's explore BERT in more detail:

1. Pre-training: BERT is pre-trained on a large corpus of unlabeled text data using a masked language modeling objective. During pre-training, BERT learns to predict masked or randomly replaced words within a sentence based on the surrounding context. This masked language modeling task enables BERT to capture deep contextual representations and learn rich word embeddings.
2. Transformer Architecture: BERT is built upon the Transformer architecture, which employs self-attention mechanisms to capture contextual dependencies. It consists of a stack of transformer encoder layers. Each layer has multiple self-attention heads and feed-forward neural networks. The transformer architecture enables BERT to efficiently model long-range dependencies and capture fine-grained contextual information.
3. Bidirectional Context: Unlike traditional language models that process text in a left-to-right or right-to-left manner, BERT is designed to be bidirectional. It takes advantage of both left and right contexts during pre-training, allowing it to capture a more comprehensive understanding of word meanings and relationships. This bidirectional context is achieved by utilizing masked language modeling and next sentence prediction tasks during pre-training.
4. Fine-tuning: After pre-training, BERT can be fine-tuned on specific downstream NLP tasks such as text classification, named entity recognition, question answering, and sentiment analysis. During fine-tuning, BERT is further trained on labeled task-specific datasets, adapting its pre-trained representations to the target task. Fine-tuning BERT often involves adding task-specific layers on top of the pre-trained BERT model.
5. Contextual Word Embeddings: One of the key contributions of BERT is its ability to generate contextual word embeddings. Traditional word embeddings such as word2vec or GloVe provide static representations of words, whereas BERT produces dynamic word embeddings that capture the context in which the words appear. The contextual embeddings produced by BERT have been shown to significantly improve performance on a wide range of NLP tasks.
6. Language Model Pre-training: In addition to masked language modeling, BERT also utilizes a next sentence prediction task during pre-training. This task involves predicting whether two sentences appear consecutively or not in the training data. By incorporating this objective, BERT learns to understand the relationships between sentences and captures discourse-level information.

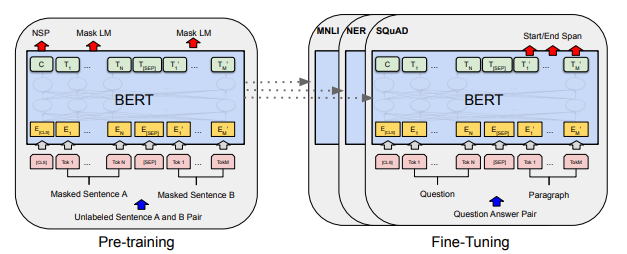
BERT has achieved state-of-the-art results on various NLP benchmarks and tasks, showcasing its effectiveness in capturing contextual information and improving the understanding of natural language. Its pre-trained representations have been widely adopted and used as feature extractors or fine-tuned for specific downstream tasks.

Figure 15 Overall pre-training and fine-tuning procedures for BERT

## 3.2 Background on Sentiment Analysis

### 2.2.1 TF -IDF

Frequency – Inverse Document Frequency (TF-IDF) is a popular statistical technique used in natural language processing and information retrieval. Its purpose is to assess the significance of a term within a specific document in relation to a collection of documents, known as a corpus. To accomplish this, TF-IDF employs a process called text vectorization, which assigns importance values to words within a document.

TF-IDF derives the importance score for a word by combining two factors: Term Frequency (TF) and Inverse Document Frequency (IDF).

Term Frequency (TF) measures how frequently a term appears within a document relative to the total number of words in that document. It is calculated by dividing the number of occurrences of a term by the total word count in the document. Essentially, TF captures the local importance of a term within a specific document.

TF =

Inverse Document Frequency (IDF) evaluates the global importance of a term across the entire corpus. It quantifies how rare or common a term is among all the documents in the corpus. IDF is computed by taking the logarithm of the total number of documents in the corpus divided by the number of documents containing the term. The logarithm is used to dampen the impact of very common terms.

IDF =

By multiplying the TF and IDF values together, TF-IDF creates a composite score that represents the relative significance of a term within a document and across the corpus. This score indicates the importance of a term in distinguishing its relevance to a particular document in comparison to other documents

### 2.2.2 LSTM

LSTM is a type of RNN that solves the problem of short-term memory by having gates that learn which data is important to keep and which can be discarded. Similar to RNNs, LSTMs processes the sequence of inputs one by one. An LSTM cell that processes one input produces a hidden state which is passed to the LSTM cell that processes the next step of the sequence. Hidden states act like a memory for the neural network enabling the information from previous steps to flow through future steps.

RNN cell which calculates the output hidden state by concatenating the input and previous hidden state and passing them through a tanh function which squishes the values to be always between -1 and 1, therefore as time passes the effect of inputs at the beginning of the sequence begin to vanish which is referred to as the short-term memory problem.

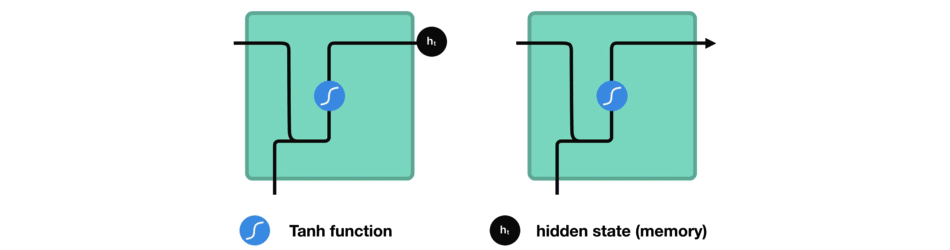


Figure 17 Flow of Hidden State through RNN.

LSTMs address the issue of short-term memory by incorporating three gates and a cell state, which enable the control of information flow and prioritize the retention of the most significant information rather than relying solely on the information at the end of the sequence. The cell state plays a crucial role in carrying relevant information throughout the entire sequence. The gates within the LSTM cells regulate the addition or removal of information from the cell state as it traverses through the network.

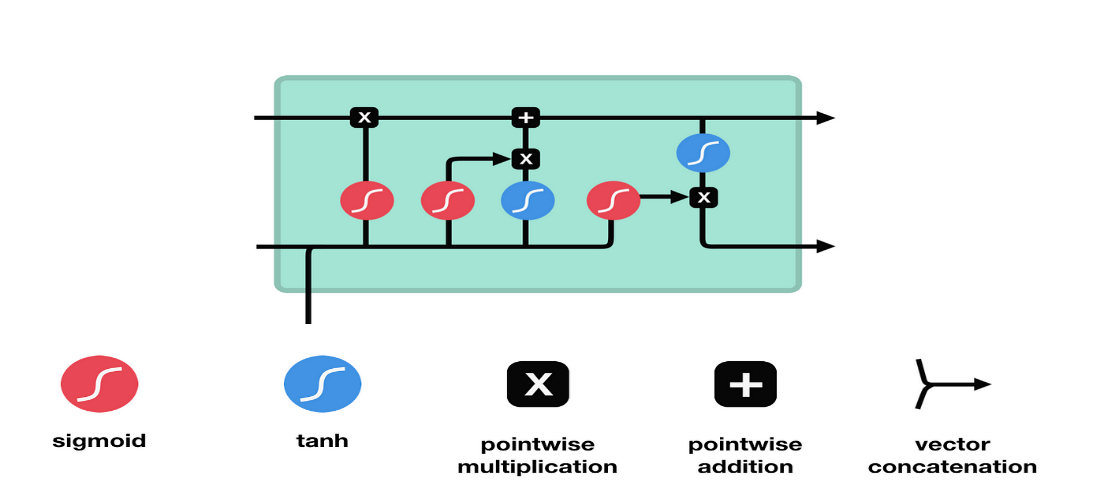


Figure 18 LSTM Cell

The forget gate plays a crucial role in determining whether information should be retained or discarded. It takes the concatenated input and previous hidden state as inputs and passes them through a sigmoid function. The output of the forget gate ranges between 0 and 1, with values closer to 0 indicating information to be forgotten and values closer to 1 indicating information to be kept.

The input gate is responsible for calculating the new cell state in conjunction with the output of the forget gate. Firstly, the concatenated hidden state and current input are fed into a sigmoid function, similar to the forget gate. Then, the concatenated hidden state and current input are passed through a hyperbolic tangent (tanh) function, which helps regulate the network's values. The output of the tanh function is multiplied by the output of the sigmoid function, with the sigmoid output determining the important information to retain from the tanh output.

The new cell state is calculated by performing point-wise multiplication of the previous cell state with the output of the forget gate, followed by point-wise addition of the result with the output of the input gate. This update process enables the neural network to adapt the cell state to new values that it deems relevant for the task at hand.

The output gate determines the next hidden state. Firstly, the previous hidden state concatenated with the current input is passed through a sigmoid function. Next, the new cell state is fed into a tanh function. The output of the tanh function is multiplied by the output of the sigmoid function to determine which information should be included in the next hidden state.

In summary, the three gates (forget gate, input gate, and output gate) in an LSTM network collectively control the flow of information. The forget gate decides what information to retain or discard, the input gate calculates the new cell state based on the input and previous hidden state, and the output gate determines the next hidden state. By incorporating these mechanisms, LSTMs are able to mitigate the short-term memory problem and effectively retain and utilize important information throughout a sequence.

### 2.2.3 Word Embedding

Word embeddings are numerical representations of words that capture their meaning, context within a document, and semantic relationships. The most widely used techniques for generating word embeddings are Word2Vec and GloVe. These methods excel at capturing the analogies between words, such as the famous "king is to queen as man is to woman" example. By performing arithmetic operations on word vectors, like subtracting the vector for "man" and adding the vector for "woman" to the vector for "king," we obtain a vector that closely aligns with the word vector for "queen."

There are two main approaches to learning word vectors: Global Matrix Factorization methods and Local Context Window methods. Each of these models has its own limitations, which can result in unsatisfactory performance if used independently.

### 2.2.3 GloVe (Global Vectors)

GloVe model is an open-source project that was developed at Stanford . It is an unsupervised learning algorithm that is used to obtain vector representations for words by combining the previous two methods.

GloVe uses matrix factorization of term-term frequency matrices, which represent co-occurrences between words as a large two-dimensional matrix where rows and columns are enumerated unique tokens in the corpus, and each entry represents how often the column term appears in the context of the row term. However, this matrix should be symmetric since the relation goes both ways, meaning that if word i appears in the context of word j, then word j must appear in the context of word i. The authors of GloVe found that using raw co-occurrences was flawed, so they used co-occurrence probability ratios instead to remove noise terms that were not related to both words. They explained this in a more detailed example in their paper. Then, they attempted to design a function that maps word vectors to ratios of co-occurrence probabilities. The purpose of this function is to discriminate any two given word vectors with the help of their context vectors. The authors then incorporate this into a least-squares regression problem with the following objective function to be minimized:

Where V is the size of the vocabulary, Xij tabulate the number of times word j occurs in the context of word i, Wi is the vector of center word i, Wk is the vector of context word k, bi is the bias term for word i, bk is the bias term for word k and Xik is the number of times word k appears in the context of word i. f is a weighting function that is used to tune our objective function so as to obey three main properties; to have a limit of 0 as x goes to 0, to be non-decreasing so as not to overweight rare co-occurrences, and to be relatively small for large values of x to not overweight frequent co-occurrences.



This expression ensures that f has values between 0 and 1, ɑ is a tuned parameter that was chosen to be equal ¾ with no real intuition behind it.

GloVe outperforms word2vec and SVD, which are local context methods, in several tasks as word analogy, word similarity and named entity recognition since it captures the global statistics of a corpus using its global objective function in addition to obtaining co-occurrence statistics using a context window over the corpus.

## 3.3 Background on Text to speech

### 2.3.1 Signal Windowing

**What is Windowing?**

The audio signal is divided into overlapping frames of a fixed duration. Each frame typically consists of a few milliseconds of the audio signal. To reduce artifacts caused by abrupt changes at the edges of the frames, a window function is applied to each frame. The window function tapers the frame's amplitude smoothly towards zero at its edges.

**Why use Windowing?**

Windowing is used to overcome the occurrence of **Spectral Leakage**, which occurs when the endpoints of an audio signal are **discontinuous** in the frequency domain, because they’re not an integer number of periods.

A picture containing text, line, font, handwriting

Description automatically generatedThese discontinuities appear as **high-frequency** components in the frequency domain that are not present in the original signal, and leak in other higher frequencies that occur, hence the name **Spectral Leakage**.

**A picture containing line, plot, diagram, text

Description automatically generatedWindowing using Hann Window**

A picture containing font, text, white, line

Description automatically generated

Applying window fn. to signal formula:

Sw(k)= S(k) .w(k),  k=1... K

A picture containing line, text, plot, diagram

Description automatically generated

### 2.3.2 Short-Time Fourier Transform (STFT)

Why do we need STFT? What is wrong with normal Fourier transformation?

The issue with normal Fourier transform is that we know what, but we don’t know when; we know when.

A picture containing diagram, plot, design

Description automatically generated

As we can see in the figure above, the normal Fourier transform basically performs a histogram showing the count of each frequency in our signal, but for our project we need more than that, we need to know when does each frequency appear, for better feature extraction and to generate the sound from text efficiently.

The proposed solution is to take smaller segments of time from the signal, apply Normal FT to it and then append those signals together. Below is how STFT is computed.

1. Windowing: The audio signal is divided into overlapping frames of a fixed duration. Each frame typically consists of a few milliseconds of the audio signal. To reduce artifacts caused by abrupt changes at the edges of the frames, a window function is applied to each frame. The window function tapers the frame's amplitude smoothly towards zero at its edges.
2. Fourier Transform: Once the frames are windowed, a Fourier transform is applied to each frame. The Fourier transform converts the time-domain signal into the frequency domain, representing the amplitude and phase of various frequency components present in the frame. The most common algorithm used for computing the Fourier transform is the Fast Fourier Transform (FFT), which efficiently computes the transform.
3. Magnitude Calculation: The result of the Fourier transform is a complex-valued representation of the signal. However, for most audio analysis tasks, we are primarily interested in the magnitude of the frequency components rather than their phase. Therefore, the magnitude spectrum is computed by taking the absolute value of the complex-valued spectrum.
4. Spectrogram Construction: The magnitude spectra obtained from each frame are typically stacked together to form a 2D matrix called a spectrogram. The x-axis represents time, which corresponds to the frames, and the y-axis represents frequency bins. The intensity of each spectrogram element represents the magnitude of the frequency component at a particular time and frequency.
5. Overlap and Hop Size: To capture the temporal evolution of the signal, adjacent frames typically overlap with each other. The amount of overlap is determined by the hop size, which refers to the number of samples by which the analysis window is shifted between consecutive frames. Commonly used hop sizes are 50% or 75% of the window size. Overlapping frames help provide a smoother transition and better time resolution in the spectrogram.

**Why do we need to overlap?**

After applying framing, and windowing (explained in the next section) on the processed signal, the endpoint of the de-framed signal suffers from information loss. This is overcome by **overlapping** the frames so information loss can be minimized.

A blue sound wave

Description automatically generated with low confidence

### 2.3.3 Mel-Spectrogram

A mel-spectrogram is a visual representation of the frequency content of an audio signal over time. To generate a mel-spectrogram we need few steps

1. Short-Time Fourier Transform (STFT): The first step in generating a mel-spectrogram is to divide the audio signal into short overlapping frames. Each frame typically consists of a few milliseconds of audio. The STFT is then applied to each frame, which involves computing the Fourier transform of the frame to obtain its frequency content.
2. Power Spectrum: The STFT produces a complex-valued spectrogram, which contains both magnitude and phase information. However, for mel-spectrogram computation, we are primarily interested in the magnitude information. The magnitude spectrogram is obtained by calculating the element-wise magnitude of the complex spectrogram.
3. Mel Filterbanks: The human auditory system does not perceive sound in a linear frequency scale but rather in a logarithmic scale. Mel filterbanks are designed to mimic this logarithmic perception. A set of triangular filters is applied to the magnitude spectrogram, where each filter captures a specific range of frequencies. The filters are evenly spaced in the mel scale, which maps the frequency axis to a perceptually relevant scale.
4. Log Compression: After applying the mel filterbanks, the magnitudes in each filter's output are summed. To compress the dynamic range and emphasize smaller magnitudes, a logarithm operation (typically base 10) is applied to the filterbank outputs.
5. Normalization: It is common to normalize the mel-spectrogram values to improve the training stability and convergence of the models. This can be done by subtracting the mean and dividing by the standard deviation across the whole spectrogram or a smaller window of frames.

A picture containing screenshot, colorfulness, text

Description automatically generatedThe resulting mel-spectrogram is a 2D representation of the audio signal, where the x-axis represents time, and the y-axis represents frequency. Each pixel in the mel-spectrogram represents the magnitude of a specific frequency component at a particular time. Higher values indicate a stronger presence of that frequency component.

We used mel-spectrogram since humans perceive frequency logarithmically; the way we perceive pitch is nonlinear, it doesn’t depend on the difference in frequency.

### 2.3.4 Mel Scale

Frequencies in the frequency domain are converted according to the Mel Scale, which is a scale used to match the human ear perception, since it doesn’t perceive frequencies linearly. For example, we can easily tell the difference between 500 and 1000 Hz, but we will hardly be able to tell a difference between 10,000 and 10,500 Hz, even though the distance between the two pairs is the same.

{Wav2lip CNN}

### A picture containing line, diagram, plot, parallel Description automatically generated2.3.5 Griffin Lim Algorithm

The Griffin-Lim algorithm is an iterative phase reconstruction algorithm used for estimating the phase information of a complex-valued spectrogram. It is commonly used in speech and audio signal processing to convert magnitude spectrograms back into time-domain waveforms. Here's a detailed explanation of the Griffin-Lim algorithm:

1. Initialization: The algorithm starts with an initial estimate of the complex-valued spectrogram, which only contains the magnitude information. The phase values are randomly initialized or set to zero.
2. Iterative Estimation: The algorithm alternates between two steps: estimation and reconstruction.
3. Estimation Step: In this step, the algorithm estimates the phase of the complex spectrogram by combining the magnitude information from the original spectrogram with the phase information obtained from the previous iteration. This is done by taking the element-wise product (Hadamard product) of the complex-valued spectrogram's magnitude and the complex exponential of the previous phase estimate:

phase = spectrogram\_magnitude \* exp(i \* previous\_phase)

1. Reconstruction Step: In this step, the estimated complex-valued spectrogram with the updated phase is transformed back into the time domain using an inverse Fourier transform. This results in a time-domain waveform estimate.
2. Magnitude Restriction: To maintain consistency with the given magnitude spectrogram, the algorithm modifies the amplitude of the reconstructed waveform by scaling it according to the original magnitude spectrogram. This ensures that the reconstructed waveform has the desired magnitude characteristics.
3. Iteration: Steps 2 and 3 are repeated for a fixed number of iterations or until convergence is achieved. The algorithm updates the phase estimate iteratively, refining it with each iteration.
4. Final Reconstruction: Once the algorithm completes the desired number of iterations, the final phase estimate is used in the last reconstruction step to obtain the reconstructed time-domain waveform.

The Griffin-Lim algorithm assumes that the magnitude spectrogram contains sufficient information to reconstruct a reasonable approximation of the original time-domain waveform.

However, it does not guarantee an exact reconstruction, and some details may be lost in the process. The algorithm can be sensitive to noise and may introduce artifacts in the reconstructed waveform.

We use the griffin Lim algorithm to compute the complex part of the signal since we computed the power of the signal so the complex part of the signal was removed, we needed to find it again.

## 3.4 Background on Avatar Generation

sasas

## 3.5 Comparative Study of Previous Work

The literature review section offers a comprehensive overview of the existing research and developments in the fields of news summarization, text-to-speech conversion, emotion synthesis, genre classification, speech-driven animation, and avatar creation. By examining relevant studies, methodologies, and approaches, we gain valuable insights into the current state of the field and identify the gaps that our research aims to address.

### 3.5.1 Summarization:

Researchers have made significant strides in news summarization techniques, aiming to condense lengthy articles into concise summaries. Extractive and abstractive summarization approaches have been explored extensively. Studies by Nenkova and McKeown (2011) have demonstrated the efficacy of using linguistic and discourse features for extractive summarization. Furthermore, the advent of deep learning and natural language processing (NLP) has led to the development of more sophisticated models like Transformer-based architectures (Vaswani et al., 2017), which have shown promising results in generating abstractive summaries.

### 3.5.2 Text-to-Speech:

The field of text-to-speech conversion has undergone significant advancements, revolutionizing how textual content is consumed. Deep learning models such as WaveNet (van den Oord et al., 2016) have demonstrated exceptional performance in generating natural and human-like speech from text. These models employ neural networks to generate speech waveforms, capturing the nuances of human speech. This technology has enabled the conversion of news articles into spoken form, providing an alternative means of consuming news content.

### 3.5.3 Sentiment Analysis:

Although still an emerging field, emotion synthesis in text-to-speech conversion has garnered significant interest. Researchers have explored models like the Emotional Neural TTS (ENTTS) model proposed by Han et al. (2019), which aim to inject emotional cues into synthesized speech. By analyzing the sentiment and emotional tone of the text, these models generate speech with appropriate emotional expressions, enhancing the engagement and impact of news consumption. The ability to effectively convey emotions through synthesized speech adds a new dimension to news delivery and further connects users to the content.

### 3.5.4 Genre Classification:

Genre classification of news articles is crucial for personalized news delivery. Researchers have employed various machine learning techniques, including Support Vector Machines (SVM) and deep neural networks, to automatically classify news articles into different genres. Noteworthy research by Denecke et al. (2008) has explored the use of textual and structural features for accurate genre classification. Recent studies have leveraged deep learning models such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to improve the accuracy and efficiency of genre classification, enabling more personalized and targeted news consumption experiences.

### 3.5.5 Avatar Generation :

In recent years, avatar creation has gained attention as a means to enhance user engagement and personalization. Avatars are virtual representations of individuals or characters that can simulate human-like appearance and behavior. In the context of news consumption, avatars can serve as narrators, bringing the news content to life. Speech-driven animation techniques have emerged, enabling avatars to synchronize their facial expressions and gestures with the synthesized speech. This approach, as demonstrated in research by Cassell et al. (2001) and Cao et al. (2019), enhances the immersive nature of news consumption and provides a more engaging and interactive experience for users.

In summary, the literature review provides valuable insights into the significant advancements in news summarization, text-to-speech conversion, emotion synthesis, genre classification, speech-driven animation, and avatar creation. By combining these components, our research aims to develop an interactive news summarizer system that incorporates avatar narration, text-to-speech conversion, emotion synthesis, and genre classification. This system aims to enhance user engagement, accessibility, and personalization in news consumption, offering a more immersive and captivating experience for users.

## 3.6 Implemented Approach

3.6.1 Summarization

3.6.2 Text to Speech

3.6.3 Sentiment Analysis

3.6.4 Avatar Generation

# Hanakhod meno le ba2y el chapter

## 3.2 Evaluation Metrics:

In the context of our project, we have developed multiple models to tackle various aspects of news summarization and presentation. Each model requires specific evaluation metrics to assess its performance and effectiveness.

*News Summarization:*

For news summarization, the commonly used evaluation metric is the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score. ROUGE measures the overlap between the generated summary and a set of reference summaries using algorithms such as ROUGE-N (n-gram overlap), ROUGE-L (longest common subsequence), and ROUGE-S (skip-bigram overlap). Higher ROUGE scores indicate better alignment between the generated summary and the references, reflecting the effectiveness of the summarization model in capturing the key information from the source article.

*Text-to-Speech Conversion:*

In text-to-speech conversion, the Mean Opinion Score (MOS) is a subjective rating provided by human listeners to assess the naturalness and quality of synthesized speech. Listeners rate the synthesized speech samples on factors such as naturalness, intelligibility, and overall preference. MOS provides insights into how well the text-to-speech model can generate speech that sounds human-like and is pleasant to listen to. Additionally, metrics like Word Error Rate (WER) can be used to evaluate the accuracy of speech synthesis in terms of correctly reproducing the input text.

*Emotion Synthesis:*

In text-to-emotion synthesis, the performance of the model can be evaluated using precision, recall, and F1-Score metrics. Precision measures the accuracy of correctly classifying emotional instances, such as sadness and joy, while avoiding false positives. Recall assesses the model's ability to identify all actual emotional instances, including those associated with sadness and joy, thereby avoiding false negatives. F1-Score provides a balanced evaluation by combining precision and recall into a single metric, offering an overall measure of the model's accuracy in classifying and expressing emotions, including the nuanced expressions of sadness and joy. These metrics allow us to comprehensively assess the effectiveness of the text-to-emotion synthesis model in accurately capturing and conveying a range of emotional states, including the specific emotions of sadness and joy, contributing to the creation of emotionally engaging news content.

*Genre Classification:*

In news genre classification, metrics such as Accuracy, Precision, Recall, and F1-Score are commonly employed. These metrics assess the model's ability to correctly classify news articles into predefined genres (e.g., sports, politics, entertainment). Accuracy measures the overall correctness of the model's predictions, while Precision and Recall provide insights into the model's ability to accurately classify positive and negative instances of each genre. F1-Score combines Precision and Recall into a single metric, providing a balanced assessment of the model's performance.

*Avatar Generation :*

In avatar generation, two commonly used evaluation metrics are LSE-D (Longest Subsequence Error - Duration) and LSE-C (Longest Subsequence Error - Coordinate). LSE-D measures the dissimilarity in movement and duration between the generated avatar animation and the reference animation, while LSE-C focuses on the dissimilarity in coordinate positions. Lower scores for both metrics indicate a higher level of accuracy and similarity in lip movements, ensuring the lip-syncing capability of the avatar generation model. These metrics provide quantitative measures of the quality and alignment between the generated avatar's lip movements and the desired targets, ensuring visually convincing and realistic synchronization

{FID}

## 3.3 Real-World Application

News summarization, text-to-speech conversion, emotion synthesis, genre classification, and avatar creation technologies have significant real-world applications beyond the realm of news consumption. For instance, these technologies can be utilized in accessibility services, helping individuals with visual impairments, or reading difficulties to access news content. By converting news articles into spoken form and providing additional visual cues through avatars, these technologies improve the accessibility and inclusivity of news information. Moreover, educational platforms can leverage these tools to enhance learning experiences by providing interactive and engaging content delivery. Virtual assistants and conversational agents can benefit from the integration of these technologies to deliver personalized news updates and engage users in natural and immersive conversations. Furthermore, these advancements have the potential to revolutionize the entertainment industry by enabling the creation of virtual news presenters, interactive storytelling experiences, and emotionally expressive characters in virtual environments. Exploring these diverse applications expands the scope and impact of the research and highlights the practical relevance of the proposed systems beyond the academic setting.

By delving deeper into the evaluation metrics, user experience, and real-world applications, your literature review will provide a comprehensive understanding of the research landscape and highlight the significance and potential impact of your project.

# Chapter 4 System Design and Architecture

{FULL SYSTEM ARCHIttrcture

## 4.1 Overview and Assumptions

## 4.2 System Architecture

### 4.2.1 Block Diagram

## 4.3 Summarizer

## 4.4 Text-to-Emotion

### 4.4.1 Input

The input of the text-to-emotion model is a text sequence containing information or dialogue that aims to convey specific emotions. This text serves as the primary input for the model, which processes and analyzes it to generate synthesized speech or other outputs that express the intended emotional content. The model utilizes NLP techniques, emotion classification algorithms, and machine learning to capture and understand the emotional nuances present in the input text.

### 4.4.2 Input dataset

The text-to-emotion model is typically trained on a dataset such as Twitter Sentiment, which consists of a collection of tweets annotated with corresponding emotions or sentiment labels. This dataset provides a valuable resource for training the model to recognize and generate emotional responses based on textual inputs. By leveraging the Twitter Sentiment dataset, the model can learn patterns and associations between specific words, phrases, and emotions, enabling it to accurately predict and express emotions in response to various text inputs. The use of such a dataset enhances the model's ability to capture the nuances and diversity of emotions expressed in social media conversations and facilitates the development of a robust and effective text-to-emotion synthesis system.

### 4.4.3 Pre-processing

Data cleaning: Removing irrelevant or noisy data such as URLs, special characters, and excessive punctuation.

Stop word removal: Eliminating common articles, pronouns, and other stop words to focus on more meaningful content.

Stemming or lemmatization: Applying techniques to normalize words and reduce variations.

Balancing the dataset: Addressing class imbalance issues by ensuring an equal representation of different emotions or sentiment labels.

Train-validation-test split: Dividing the preprocessed dataset into training, validation, and testing sets for model training and evaluation.

Tokenization: Splitting the text into individual words or tokens for further analysis and processing.

Label-Encoding:

As part of the preprocessing steps, the categorical features representing emotions in the Twitter Sentiment dataset are transformed using label encoding. Label encoding is a technique that assigns a unique numerical label to each distinct emotion category. This conversion enables the text-to-emotion model to process and analyze the emotion data more effectively during training. By mapping emotions to numerical labels, the model can better understand and capture the underlying patterns and relationships between different emotional states. The label encoding process facilitates the conversion of categorical emotion features into a format that can be easily fed into the architecture of the text-to-emotion model, ultimately enhancing its ability to generate accurate and appropriate emotional responses based on the given input text.

These preprocessing steps help refine the dataset, improve data quality, and enhance the effectiveness of the text-to-emotion model architecture in recognizing and generating emotions based on the given text inputs.

### 4.4.4 Tokenization and word embedding

#### 5.2.4.1 Tokenization

Tokenization is a crucial step in natural language processing that involves splitting text into individual tokens or words. When performing tokenization, a dictionary of words is created. This dictionary, also known as a vocabulary, contains all the unique words present in the training data. Each word is assigned a unique index or token ID.

During tokenization, the text is divided into tokens based on specific rules or techniques. For example, tokens can be created by splitting the text at whitespace characters, punctuation marks, or by applying more sophisticated algorithms such as word-based or subword-based tokenization.

The creation of a dictionary of words allows the text-to-emotion model to represent words as numerical values. This enables the model to process and analyze the text using mathematical operations. Each word in the input text is replaced with its corresponding token ID from the vocabulary, forming a sequence of numerical values that the model can understand and process.

Additionally, tokenization helps in handling the issue of rare or unknown words. Unknown words are typically assigned a special token, such as "UNK," during tokenization. This ensures that even if the model encounters words that were not present in the training data, it can still represent them using the "UNK" token, maintaining a consistent vocabulary and facilitating further processing.

In summary, tokenization plays a crucial role in creating a dictionary of words, allowing the text-to-emotion model to represent and process text inputs as sequences of numerical tokens. It enables the model to understand the structure and meaning of the text, facilitating accurate emotion analysis and generation based on the given input.

#### 5.2.4.2 Word Embedding

Word embedding is a technique used in natural language processing (NLP) to represent words as dense vectors in a continuous vector space. These word vectors capture semantic relationships and contextual information, enabling the model to understand the meaning and similarity between different words.

One popular word embedding model is GloVe (Global Vectors for Word Representation). GloVe is trained on large corpora of text data and aims to capture the co-occurrence statistics of words. It leverages the idea that words that appear in similar contexts tend to have similar meanings.

GloVe provides pre-trained word vectors of varying dimensions and vocabulary sizes. For example, the GloVe 6B 200d variant represents words as vectors with a dimensionality of 200. The "6B" in the name refers to the fact that this variant was trained on a corpus containing 6 billion tokens.

A picture containing text, diagram, screenshot, line

Description automatically generated

Figure 19 GloVe Word Embedding

To utilize GloVe word embeddings in the text-to-emotion model, the pre-trained word vectors are loaded into the model as an embedding layer. During training or inference, each word in the input text is mapped to its corresponding word vector in the embedding layer. These word vectors provide a numerical representation of the words, capturing their semantic meaning and relationship with other words.

By leveraging GloVe word embeddings, the text-to-emotion model can benefit from the contextual information and semantic relationships embedded in the vectors. This enhances the model's ability to understand the meaning of words in the input text, improving its performance in tasks such as emotion recognition and generation.

Overall, the use of GloVe word embeddings, such as the GloVe 6B 200d variant, allows the text-to-emotion model to encode words as dense vectors, capturing their contextual information and semantic similarities. This aids the model in understanding and generating appropriate emotional responses based on the given text inputs.

### 5.2.5 Full Model Architecture

The initial layer is an "Embedding" layer that receives text encoded as integers and retrieves the corresponding embedding vector for each word. It produces a 3D tensor with dimensions (batch\_size, sequence\_length, embedding\_dim), where batch\_size represents the number of examples in the batch, sequence\_length is the length of the input sequences (229 words in this case), and embedding\_dim denotes the size of the embedding vectors (200 dimensions). The embedding layer has 2,863,600 trainable parameters.Following the Embedding layer, there are three "Bidirectional" layers that employ both forward and backward Long Short-Term Memory (LSTM) units to process the input. LSTMs are a type of recurrent neural network capable of capturing long-term dependencies in sequential data. Each bidirectional layer generates a 3D tensor with dimensions (batch\_size, sequence\_length, units), where units represents the number of LSTM units in the layer. In this instance, the first bidirectional layer has 512 units, the second has 256 units, and the third has 256 units. These layers involve a substantial number of trainable parameters due to the complexity of LSTM models and their internal weights.The final layer is a "Dense" layer that applies a linear transformation to the input, producing the output. The output has a shape of (batch\_size, 6), indicating the presence of 6 classes. The dense layer consists of 1,542 trainable parameters.In total, the model comprises 4,851,702 trainable parameters and 2,863,600 non-trainable parameters.

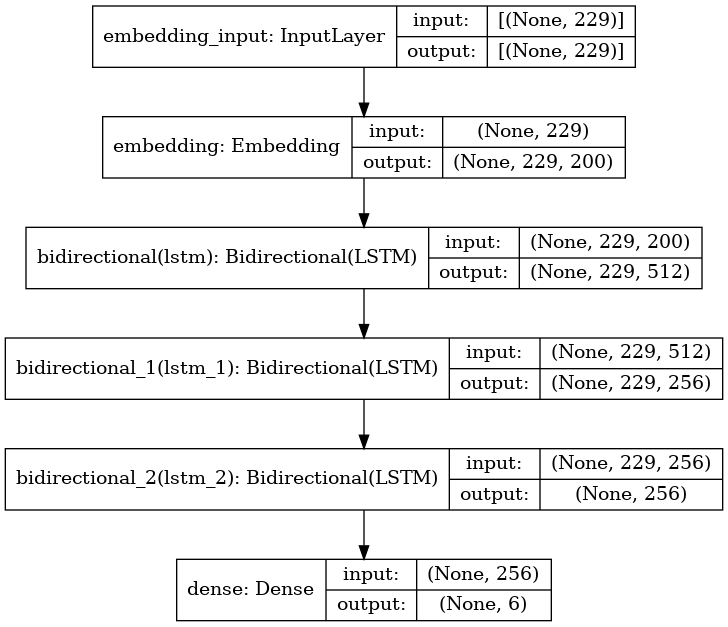


Figure 20 Text-to-Emotion Model Architecture

### 5.2.6 Using LSTM Over RNN

LSTM (Long Short-Term Memory) units were chosen over traditional RNN (Recurrent Neural Network) units in the model due to their ability to address certain limitations associated with gradient descent and the vanishing/exploding gradient problem.

|  |  |
| --- | --- |
| Text | Predicted class /probability |
| last two years were very hard due to the pandemic I was depressed. | sadness: 0.4253944158554077 |
| The last two years were very hard due to the pandemic I was depressed, but now my life is very good. I am living my dreams and I am very happy | joy: 0.867871105670929 |

One significant advantage of LSTM is its capability to mitigate the vanishing gradient problem, which occurs when the gradients used for updating the model's parameters become exponentially small as they propagate through time. By introducing a forget gate in the LSTM architecture, the model can selectively retain or discard information from the past, allowing for the preservation of important information over longer sequences. This helps prevent the loss of crucial context during training and enables better long-term memory retention in the network.

Furthermore, the presence of the forget gate allows the LSTM to regulate the flow of information within the cell. It determines which information should be forgotten and what new information should be stored in the cell state. This gating mechanism enhances the model's ability to capture long-term dependencies and adaptively update the cell state based on the input and the previous hidden state.

In summary, by utilizing LSTM units instead of traditional RNN units, the text-to-emotion model can overcome the vanishing gradient problem, better capture long-term dependencies, and leverage the forget gate mechanism to update and control the information flow within the LSTM cells. These aspects contribute to the model's improved learning capabilities and its ability to generate more accurate and contextually rich emotional responses based on the input text.

To elaborate the advantage of the forget gate of the LSTM we tested our model on a text input that triggers different emotions at the end of the flow of the sentence that would give a different emotion to the sentence

### 5.2.7 Model Evaluation

The text-to-emotion model was evaluated using precision, recall, and F1 score metrics for each emotion category. Precision measures the proportion of correctly predicted instances for a specific emotion out of all instances predicted as that emotion. Recall, on the other hand, measures the proportion of correctly predicted instances for a specific emotion out of all instances that truly belong to that emotion. F1 score combines precision and recall into a single metric, providing a balanced measure of the model's performance for a particular emotion.

To calculate precision, recall, and F1 score for each emotion category, the model's predictions were compared against the ground truth labels in the evaluation dataset. True positive (TP), false positive (FP), and false negative (FN) values were computed for each emotion. TP represents the number of correctly predicted instances for the specific emotion, FP represents the number of instances incorrectly predicted as the emotion, and FN represents the number of instances missed by the model for the emotion.

Precision was calculated as TP divided by the sum of TP and FP, providing an estimate of the model's ability to accurately predict the specific emotion category. Recall was calculated as TP divided by the sum of TP and FN, indicating the model's effectiveness in identifying all instances of the specific emotion. F1 score was computed as the harmonic mean of precision and recall, providing a balanced assessment of the model's performance for the given emotion.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1-score | Support |
| Anger | 0.93 | 0.92 | 0.93 | 275 |
| Fear | 0.95 | 0.84 | 0.89 | 224 |
| Joy | 0.93 | 0.95 | 0.94 | 695 |
| Love | 0.82 | 0.82 | 0.82 | 159 |
| Sadness | 0.96 | 0.96 | 0.96 | 581 |
| Surprise | 0.74 | 0..91 | 0.82 | 66 |
| Accuracy |  |  | 0.92 | 2000 |
| Macro avg | 0.89 | 0.90 | 0.89 | 2000 |
| Weighted avg | 0.93 | 0.92 | 0.92 | 2000 |

## 5.3 News Genre Classifier

### 5.3.1 Input

The input of the news genre classifier consists of textual data extracted from news articles. This textual data typically includes the title, headline, summary, and main content of the article. The classifier takes this input text as its primary source of information for determining the genre or category of the news article.

The news genre classifier is designed to classify news articles into several predefined categories, including politics, sports, technology, entertainment, and business. The input text is analyzed by the classifier, which examines the patterns, context, and semantic information present in the text to make accurate genre predictions. By leveraging machine learning or deep learning algorithms, the classifier determines the most suitable genre for each news article, providing insights into its subject matter. The classifier's ability to accurately categorize news articles into politics, sports, technology, entertainment, or business genres allows users to efficiently navigate and access news content that aligns with their specific interests and preferences.

n summary, the input of the news genre classifier consists of preprocessed textual data extracted from news articles. This input undergoes various preprocessing steps and is represented in a numerical format suitable for the classifier model. The classifier then utilizes machine learning or deep learning algorithms to analyze the input text and predict the genre or category of the news article.

## 5.3.2 Pre-processing

The input text undergoes several preprocessing steps to prepare it for classification. These steps may include removing irrelevant characters or symbols, converting the text to lowercase, and handling special cases such as stopwords or punctuation. Additionally, the text may be tokenized to split it into individual words or subwords, allowing for more granular analysis.

Once the preprocessing is complete, the input text is represented in a numerical format suitable for the classifier model. This can be achieved through techniques such as word embedding, where words are mapped to dense vectors, or one-hot encoding, where words are represented as binary vectors indicating their presence or absence in the text.

The preprocessed and encoded input text is then fed into the news genre classifier model, which applies various machine learning or deep learning algorithms to make predictions about the genre of the news article. These algorithms analyze the patterns, context, and semantic meaning present in the input text to make accurate genre predictions.

### 5.3.3 Model Architecture

The news genre classifier shares a similar architecture with the text-to-emotion model, featuring LSTM layers and an embedding layer. These components enable the model to capture sequential dependencies and extract meaningful representations from the input text. However, the main distinction between the two models lies in the preprocessing steps required for each task.

While the text-to-emotion model focuses on understanding and predicting emotions from textual input, the news genre classifier is designed to classify news articles into different genres or categories. As a result, the preprocessing steps for the news genre classifier may involve specific techniques such as handling stopwords, punctuation, or domain-specific data cleaning procedures to enhance genre classification accuracy.

Despite the differences in preprocessing, both models leverage LSTM layers to capture long-term dependencies and an embedding layer to convert the input text into numerical representations. These shared architectural components enable the models to learn complex patterns and relationships in the input data, enhancing their ability to make accurate predictions.

By adopting a similar architecture to the text-to-emotion model, the news genre classifier benefits from the LSTM layers' ability to capture contextual information and the embedding layer's capability to represent words as dense vectors. This allows the news genre classifier to effectively classify news articles into politics, sports, technology, entertainment, or business genres based on the underlying patterns and characteristics present in the input text.

### 5.3.4 Model Evaluation

{TODO evaluate Category}

## 5.3 Text to Speech

### 5.3.1 Input

The Tacotron synthesis process involves encoding preprocessed text, passing it through the Tacotron model for inference, and generating mel-spectrograms that represent speech characteristics. These mel-spectrograms are post-processed, and then converted into time-domain waveforms using a vocoder. The synthesized waveforms capture the acoustic properties of the input text, resulting in natural-sounding speech.

### 5.3.2 Input Dataset

During the training phase of the Tacotron model, we used LJ Speech dataset.

The input consists of pairs of text and corresponding audio waveforms. These pairs are used to train the model to learn the mapping between the text input and the desired speech output.

During training, the text and audio waveform pairs are aligned such that each text instance corresponds to its corresponding audio waveform. The pairs are organized into a dataset, where each data sample consists of the text and the corresponding audio waveform.

The training input pairs are used to optimize the Tacotron model's parameters by minimizing the discrepancy between the predicted mel-spectrograms (generated from the text input) and the target mel-spectrograms (derived from the audio waveform). The model is trained using supervised learning techniques, where the model learns to generate mel-spectrograms that capture the desired speech characteristics when given input text.

### 5.3.3 Pre-processing

Data pre-processing for Tacotron training involves a few steps for both the text and the audio files

Below are the steps we made in order to optimize our Model

1. Text Processing:

* Normalization: Convert text to lowercase and remove certain punctuation.
* Tokenization: Split text into individual characters or phonemes.
* Text Cleaner: Clean text by removing specific patterns or characters.

1. Audio Processing:

* Audio Loading: Load audio files (WAV format) from the dataset.
* Audio Preprocessing: Apply pre-emphasis to the audio signal.
* Audio Normalization: Normalize audio amplitude to a target value.
* Spectrogram Computation: Compute mel-spectrograms from the audio using the Short-Time Fourier Transform (STFT).

1. Dataset Creation:

* Pair Text and Audio: Associate each text input with its corresponding audio spectrogram.
* Split into Train/Validation Sets: Divide the dataset into training and validation subsets.

1. Text Token Indexing:

* Create Vocabulary: Build a vocabulary of unique tokens (characters or phonemes).
* Assign Indices: Map each token to a unique index in the vocabulary.

1. Data Batching:

* Group Data: Group text and audio spectrograms into batches for efficient training.
* Padding: Pad sequences to have equal lengths within each batch.
* Create Masks: Generate masks to ignore padding regions during model training.

These data processing steps prepare the text and audio data for training the Tacotron model. The text is transformed into a numerical representation (token indices), and the audio is converted into mel-spectrograms. Batching and padding ensure efficient training, and masks help the model focus on relevant parts of the data.

### 5.2.5 Full Model architecture

The full Tacotron model architecture consists of several components:

1. Encoder: The encoder module takes the preprocessed text input and converts it into a high-level textual representation. It typically employs recurrent neural networks (RNNs), such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) layers, to capture the temporal dependencies in the text.
2. CBHG Module

The CBHG (Convolutional Banks and Highway Networks followed by a Bidirectional GRU) module combines convolutional filters, pooling, highway networks, and bidirectional GRU layers to extract high-level representations from sequential data, capturing frequency characteristics, reducing dimensionality, and capturing temporal dependencies.

The CBHG module is designed to learn hierarchical representations of the input sequence, capturing different levels of abstraction. It transforms the input sequence through convolutional filters, pooling, highway networks, and bidirectional GRU layers, resulting in higher-level representations that can be used for subsequent tasks, such as generating mel-spectrograms or modeling linguistic features in speech synthesis models like Tacotron.

A picture containing text, screenshot, diagram, font

Description automatically generated

1. Attention Mechanism: The attention module is used to align the encoded text representation with the generated mel-spectrograms. It calculates attention weights that indicate the importance of each text encoder output at each step of mel-spectrogram generation. This allows the model to focus on relevant parts of the text during synthesis.
2. Decoder: The decoder module takes the combined information from the attention mechanism and the previous mel-spectrogram frames as input. It generates the next frame of the mel-spectrogram, capturing the spectral characteristics of the speech. The decoder also utilizes RNN layers to model the temporal dependencies in the mel-spectrogram generation process.
3. Post-processing: The generated mel-spectrograms may undergo post-processing steps, such as smoothing or dynamic range compression, to improve their quality and naturalness.
4. Vocoder: The generated mel-spectrograms are transformed into time-domain waveforms using a vocoder. A vocoder converts the mel-spectrograms into speech waveforms by synthesizing the corresponding speech signal.

A picture containing text, screenshot, font, number

Description automatically generated

The Tacotron model architecture is designed to learn the complex mapping between text and speech by generating mel-spectrograms that capture the acoustic characteristics of the speech. It utilizes an encoder to encode the text, an attention mechanism to align the text and spectrogram, a decoder to generate the spectrogram frames, and a vocoder to synthesize the final speech waveform.

A picture containing text, diagram, plan, screenshot

Description automatically generated