Faculty of Engineering - Cairo University
Credit Hour System Programs

Computer and Comunication Engineering

CCEC

Graduation Project Report

Spring/2023

Speak News

Prepared by:

Ahmed Ehab Farghal
Youssef Kadry Hashem
Mohamed Amr Afifi
Abdallah Wael Marzouk

Supervised by:

Dr. AbdElMoniem Bayoumi

CUFE Spring/2023
CCE
Credit Hour System

Senior-2 Level
Graduation Project-2
CCEN481

Graduation Project-2

“SpeakNews”

Final Report

Submitted by:

Ahmed Ehab Farghal

Youssef Kadry Hashem

Mohamed Amr Afifi

Abdallah Wael Marzouk

Supervised by:

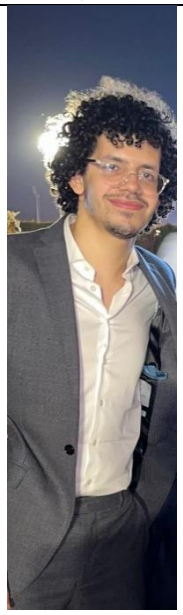
Dr. AbdElMoniem Bayoumi

Acknowledgement

Listed below are the names of the people who provided us with significant help in developing our graduation project to all we extend our sincere thanks

****

Dr. AbdElMoniem Bayoumi

<table>
<tr><td>Project Code</td><td colspan="2">GP-N481</td></tr>
<tr><td>Project Title</td><td colspan="2">SpeakNews</td></tr>
<tr><td>Keywords</td><td colspan="2">Natural Language Processing, Machine Learning, Speech recognition, sentiment analysis, avatar generation, lip syncing, Digital Signal Processing, Computer Vision, mobile application.</td></tr>
<tr><td>Students</td><td> 
Name: Ahmed Ehab Farghal
ID: 1180026
Phone: +201008536356
Email: ahmedehabb17@gmail.com</td><td> 
Name: Youssef Kadry Hashem
ID: 1180025
Phone: +201117800943
Email: yousefqadryy@gmail.com</td></tr>
<tr><td></td><td> 
Name: Mohamed Amr Afifi
ID: 1180062
Phone: +201066991087
Email: mohamed.amr.afifi@gmail.com</td><td> 
Name: Abdallah Wael Marzouk
ID: 1180376
Phone: +201030702155
Email: abdallahwm1@gmail.com</td></tr>
<tr><td>Supervisor</td><td colspan="2">Name: Dr. AbdElMoniem Bayoumi
Email:</td></tr>
<tr><td>Project summary</td><td colspan="2">Our project aims to take news consumption to a different perspective. We achieve this by having the news articles summarized and narrated by an avatar</td></tr>
</table>

Cairo University
Faculty of Engineering
Department of Computer Engineering

Speak News

A Graduation Project Report Submitted
to
Faculty of Engineering, Cairo University
in Partial Fulfillment of the requirements of the degree
of
Bachelor of Science in Computer Engineering.

Presented by:

Ahmed Ehab Farghal
Youssef Qadry Hashem
Mohamed Amr Afifi
Abdallah Wael Marzouk

Supervised by:

Dr. AbdElMoniem Bayoumi

Saturday, June 10, 2023

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.

ملخص

الأساليب التقليدية لقراءة الأخبار غالبًا ما تشعر بالملل وتستغرق وقتًا طويلاً، مما يؤدي إلى تراجع المشاركة وتحميل المعلومات الزائدة. لمعالجة هذه التحديات، نقدم نظام ملخص الأخبار التفاعلي الذي يستغل الرواية بواسطة الشخصيات الافتراضية وتقنيات تحويل النص إلى كلام. تهدف حلولنا إلى ثورة استهلاك الأخبار من خلال توفير ملخصات أخبار موجزة يمكن الاستماع إليها بسهولة أو تجربتها بصريًا من خلال الشخصيات الافتراضية.

مع التعرف على الاستياء المتزايد من تنسيقات قراءة الأخبار التقليدية والحاجة إلى استهلاك المعلومات بكفاءة، يتعامل نظامنا مع هذه المسائل مباشرة. من خلال استخدام تقنيات متقدمة لمعالجة اللغة الطبيعية وتعلم الآلة، نقوم بتوليد ملخصات أخبار شاملة تلقائيًا، من خلال تجميع المعلومات الرئيسية من المقالات.

تقدم تطبيقنا للمستخدمين تجربة محسنة لاستهلاك الأخبار من خلال توفير ملخصات أخبار موجزة من خلال الرواية بواسطة الشخصيات الافتراضية وتحويل النص إلى كلام. يمكن للمستخدمين الاستماع إلى المقالات الإخبارية بكل سهولة أو تجربتها بصريًا، متجاوزين الملل والقيود الزمنية المرتبطة بأساليب القراءة التقليدية. من خلال تجميع المعلومات الرئيسية باستخدام تقنيات متقدمة لمعالجة اللغة الطبيعية، يضمن نظامنا ملخصات دقيقة وسهلة الاستيعاب

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

List of Figures

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
MFCCs	mel-frequency cepstral coefficients
CNN	convolution neural network
RNN	recurrent neural networks
LSE-D	Lip Sync Error – Difference
LSE-C	Lip Sync Error – Confidence
FID	Fréchet Inception Distance

Contacts

Team Members

Name	Email	Phone number
Ahmed Ehab Farghal (team leader)	ahmedehabb17@gmail.com	01008536356
Youssef Kadry Hashem	yousefqadryy@gmail.com	01117800943
Mohamed Amr Afifi	mohamed.amr.afifi@gmail.com	01066991087
Abdallah Wael Marzouk	abdallahwm1@gmail.com	01030702155

Supervisor

Supervisor name	Email
Dr. AbdElMoniem Bayoumi	abdelmoniem.bayoumi@gmail.com

This page is left intentionally empty.

Chapter 1 Introduction

1.1 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.

1.2 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.

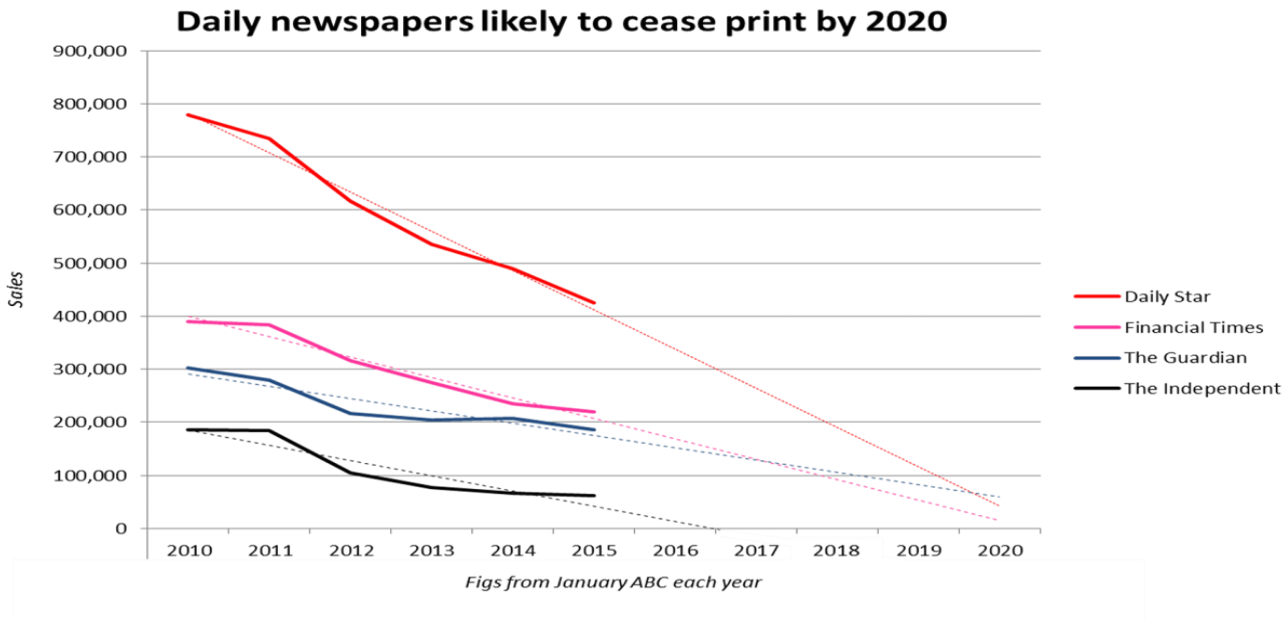


Figure 1.1 Decline in printed newspaper.

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.

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.

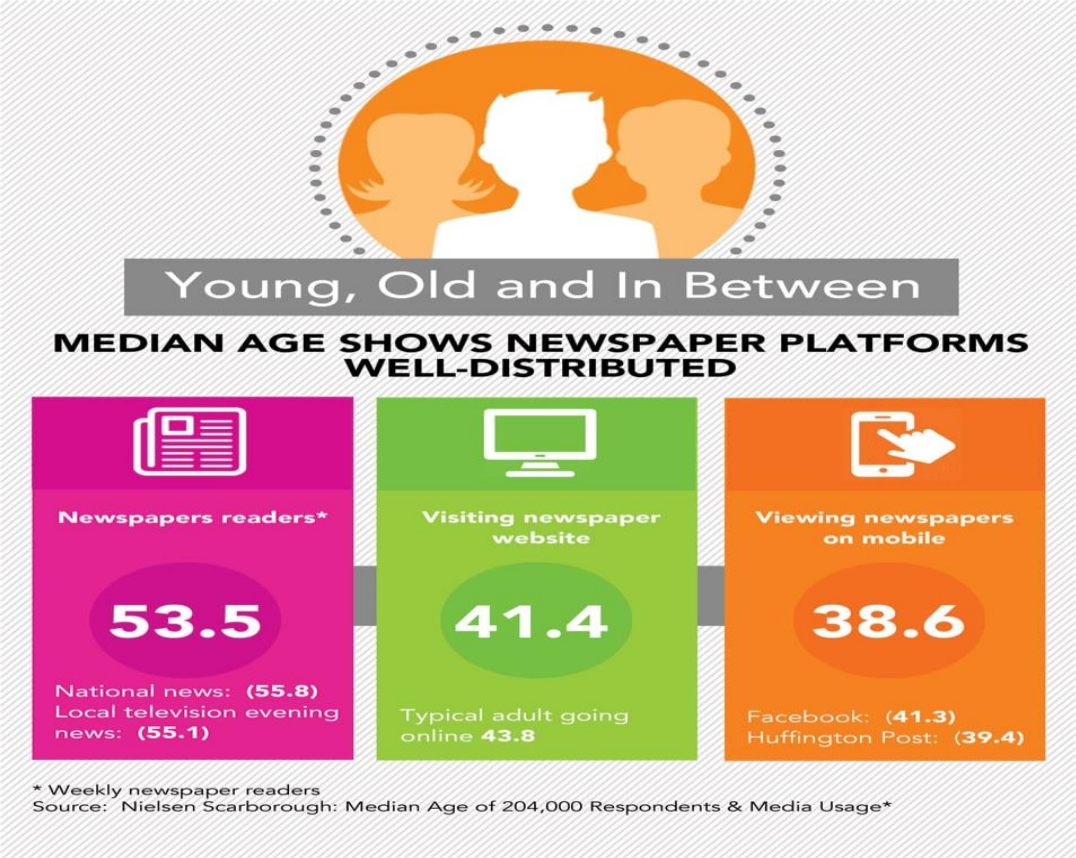


Figure 1.2 Median age for news consumption.

1.3 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.

In Chapter 1, the introduction, we will include an introduction to 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.

In chapter 2, we talk about the market visibility study. This includes the targeted customers, and a market survey. We mentioned our competitors who have similar ideas. Finally, we talked about business cases and financial analysis.

In chapter 3, 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. In addition to this, we explore 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.

In chapter 4, 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.

In chapter 5, we cover the testing setup, methodology, and schedule, including module testing and integration testing. The results obtained from testing different modules and the integrated system are discussed, emphasizing any issues or successes encountered. Furthermore, a summary of comparative results to previous work is provided, presenting tabulated and/or graphical representations along with a brief commentary. This chapter serves as a valuable resource for understanding the comprehensive testing and verification processes implemented to validate the project's functionality.

The final chapter, we conclude our project then highlight 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

The mobile application market has experienced explosive growth in recent years, catered to diverse user needs and preferences. With millions of apps available across various platforms, developers are constantly seeking innovative ways to stand out and capture the attention of users. We created a new innovative idea of a mobile application to revolutionize the news reading experience. Speak News is a news summarizer that detects the sentiment from text. In addition to this, it gives two options: option 1 is to listen to the summary's audio for hands-free consumption, or option 2 the user can watch a video of an avatar speaking the news for a more visually appealing, engaging option. With Speak News, you can read the latest news and information, whenever and wherever you are.

2.1 Target Customers

In the rapidly evolving digital landscape, where information is abundant and time is a precious resource, news consumption habits have undergone a significant transformation. As traditional methods of news reading face challenges in accommodating modern lifestyles, innovative solutions like SpeakNews have emerged to address these evolving needs. SpeakNews is a news summarizer app that aims to provide a streamlined and efficient way for users to stay informed in a time-constrained world.

Understanding the target customers of SpeakNews is vital for its successful deployment and user adoption. By identifying the specific audience that the app caters to, we can tailor its features, functionalities, and marketing strategies to effectively meet their needs. In this section, we explore the diverse range of individuals who are most likely to find value in SpeakNews and benefit from its unique offerings.

The target customers for SpeakNews encompass a variety of user segments, each with their own motivations and requirements. From news enthusiasts seeking convenience to busy professionals grappling with time constraints, the app has something to offer for everyone. Furthermore, SpeakNews extends its utility beyond these primary groups to cater to commuters, visually impaired individuals, language learners, and elderly individuals, among others.

By delving into the characteristics and preferences of these target customers, we gain valuable insights into their motivations for seeking news summaries delivered in an audio format. The ability to grasp the essence of news articles quickly, regardless of the user's location or visual capabilities, makes SpeakNews a compelling solution for a wide range of users.

1. **News Enthusiasts**: SpeakNews can cater to individuals who have a keen interest in staying updated with the latest news but may not have enough time to read through lengthy articles. These customers value convenience and efficiency in accessing news content.

2. **Busy Professionals**: Professionals with demanding schedules, such as business executives, entrepreneurs, or lawyers, often struggle to find time to consume news. SpeakNews can be a valuable tool for them to quickly grasp the main points of news articles during their busy routines.

3. **Commuters**: People who commute regularly, whether by public transportation or car, may prefer consuming news in audio format due to the constraints of reading while on the move. SpeakNews can provide them with a hands-free news experience, allowing them to catch up on the latest updates while commuting.

4. **Visually Impaired Individuals**: SpeakNews can be a particularly useful tool for visually impaired individuals who rely on audio-based content. The app's text-to-speech feature can enable them to access news articles effectively and independently, expanding their access to information.

5. **Language Learners**: Language learners who want to practice their listening skills and improve their vocabulary in a specific language can benefit from SpeakNews. By summarizing news articles and converting them into spoken words, the app can offer an immersive language learning experience.

6. **Elderly Individuals**: Some older adults may prefer spoken news content over reading due to vision-related challenges or limited familiarity with digital interfaces. SpeakNews can provide them with a user-friendly platform to access news and stay connected with current events.

The targeted customers benefit from this project in the following ways:

1. **Time-saving Convenience**: Target customers, such as busy professionals and individuals with demanding schedules, will benefit from SpeakNews' ability to deliver news summaries. By condensing lengthy articles into concise and easily digestible summaries, the app saves users valuable time, allowing them to stay informed without investing significant reading hours.

2. **Accessibility and Inclusivity**: SpeakNews caters to a diverse audience, including visually impaired individuals and elderly users. Through its text-to-speech functionality, the app provides an accessible and inclusive platform for news consumption. Users with visual limitations can now effortlessly listen to news articles, ensuring equal access to information and bridging the digital divide.

3. **On-the-go News Consumption**: Commuters and individuals who are frequently on the move can benefit from SpeakNews' audio-based news delivery. By converting news summaries into spoken words, the app enables users to consume news hands-free, making their daily commutes more productive and engaging.

4. **Language Learning Tool**: Language learners seeking to improve their listening skills and expand their vocabulary will find SpeakNews valuable. By offering news summaries in various languages, the app facilitates language immersion and

provides a practical context for learning, helping users enhance their language proficiency.

5. **Emotional Awareness and Analysis**: SpeakNews incorporates text sentiment analysis, enabling users to gain insights into the emotional tone of news articles. This feature benefits users who want to understand the sentiment distribution of news topics, track trends, or analyze public opinion on specific subjects.

6. **Personalization and Control**: SpeakNews allows users to customize their news preferences, ensuring that they receive summaries aligned with their interests. By empowering users to select specific news categories or prioritize certain topics, the app delivers a personalized news experience, tailored to individual preferences and enhancing user engagement.

2.2 Market Survey

There are a few potential competitors. The list of competitors are as follows:

2.2.1 NewsBlaze:

NewsBlaze is the closest idea for our project. It is a news website that offers audio and video news summaries. It shares the concept of providing audio and video content alongside news summaries. However, it doesn't provide the sentiment analysis part.

2.2.2 Google News:

Google News is a popular news aggregator that provides personalized news summaries based on user preferences and interests. However, it doesn't offer sentiment analysis nor audio/video options which means a weak competitor.

2.2.3 Apple News:

Apple News is another news aggregation platform available on Apple devices. It selects news articles from various sources and offers personalized news recommendations. The sentiment analysis and audio/video option are also not met.

2.2.4 Summly:

Summly, which was acquired by Yahoo in 2013, aimed to provide short news summaries optimized for mobile devices. Summarization of the news article is met. However, the other options are also not met.

2.2.5 IBM Watson Tone Analyzer:

IBM Watson Tone Analyzer is an API service that provides sentiment analysis and tone detection for text. The summarization and audio/video options are not met.

Below is a table comparing all the competitors with the services they provide against our product:

Competitor	Text Summarization	Text to Speech	Text Sentiment Analysis	Avatar Generation
NewsBlaze	Yes	Yes	No	Yes
Google News	Yes	No	No	No
Apple News	Yes	No	No	No
Summly	Yes	No	No	No
IBM Watson Tone Analyzer	No	No	Yes	No

Table 2.1 Competitors of our product

As you can see there is no one-stop service that provides our solution end-to-end Which gives us a great competitive advantage in the Market.

2.3 Business Case and Financial Analysis

Market Size and Growth:

The market size and growth of the news consumption market, as well as the avatar generation industry, are both significant indicators of the potential success and viability of the SpeakNews project.

1. News Consumption Market: The news consumption market has witnessed remarkable growth in recent years, driven by the increasing availability of digital content and the rise of mobile devices. According to market research, the global digital news market was valued at over $100 billion in 2020 and is projected to experience steady growth in the coming years. Factors such as the rapid expansion of internet penetration, the proliferation of smartphones, and the growing demand for personalized news experiences contribute to this market growth.

Moreover, the shift towards convenience and time-efficiency in news consumption has led to the emergence of news summarization and aggregation services like SpeakNews. As users seek ways to stay informed amidst their busy schedules, the demand for solutions that provide condensed and easily digestible news summaries is expected to rise. This trend creates a favorable market environment for SpeakNews, positioning it as a valuable tool to meet the evolving needs of news consumers.

2. Avatar Generation Industry: The avatar generation industry has gained substantial traction due to advancements in technologies such as artificial intelligence (AI) and computer vision. Avatars, digital representations of individuals that can mimic human speech and movements, have found applications in various domains, including entertainment, virtual assistants, and communication tools.

The market for avatar generation is witnessing significant growth, with projections indicating a compound annual growth rate (CAGR) of over 20% in the coming years. The increasing integration of avatars in various sectors, including advertising, e-learning, and customer service, further fuels this growth. Avatars have been proven to enhance user engagement, provide interactive experiences, and facilitate more human-like interactions in digital environments.

By leveraging avatar technology, SpeakNews can elevate the news consumption experience by generating a talking avatar that delivers audio-based news summaries. This unique feature adds an element of interactivity and visual engagement, creating a distinct user experience compared to traditional news consumption methods.

As the market size and growth of both the news consumption market and the avatar generation industry continue to expand, the SpeakNews project stands to benefit from this favorable market landscape. The demand for convenient news consumption solutions and interactive digital experiences sets a promising stage for the app's success, enabling it to tap into a growing user base and capitalize on the evolving preferences of news consumers.

Based on the market survey, we can anticipate selling the following number of user subscriptions over the next 5 years:

Year	**App Purchases**
2024	10,000
2025	60,000
2026	70,000
2027	80,000
2028	100,000
Total over 5 years	320,000

Table 2.2 Number of user subscriptions over the next 5 years

This projection is based on the following assumptions:

- The market for emotion detection and recognition is growing at a rate of 15% per year.
- Our product is the most accurate and reliable emotion detection and recognition solution on the market.
- We are able to effectively market and sell our product.

Pricing

Here is a possible pricing model for a project that is $2 per download:

Price: $2 per download from Appstore or Playstore

Free Trial:

- Offer a free trial of the project for 14 days.
- During the free trial, users can access all the project's features.
- After the free trial period, users can continue to use the project by paying the $1 per download fee.

Payment Methods:

- Credit card
- PayPal
- Google Pay
- Apple Pay

Refund Policy:

- Users can request a refund within 30 days of purchase.
- Refunds will be processed within 10 business days.

This pricing model is based on the following assumptions:

- The project is high-quality and valuable to users.
- Users are willing to pay $1 per download for the project.
- The project can generate enough revenue to cover its costs and make a profit.

Capex

- **Software development**: The development of your software product will require a significant upfront investment. This investment will cover the costs of hiring developers, purchasing software licenses, and acquiring hardware.
- **Marketing**: You will need to invest in marketing your product to potential customers. This investment could cover the costs of advertising, public relations, and events.
- **Sales**: You will need to invest in sales to generate leads and close deals. This investment could cover the costs of sales salaries, commissions, and marketing materials.

Opex

- **Software maintenance**: Once your software product is developed, you will need to maintain it. This maintenance will cover the costs of bug fixes, security updates, and feature enhancements.
- **Marketing**: You will need to continue to invest in marketing your product to potential customers. This investment could cover the costs of advertising, public relations, and events.
- **Sales**: You will need to continue to invest in sales to generate leads and close deals. This investment could cover the costs of sales salaries, commissions, and marketing materials.
- **General and administrative expenses**: These expenses include the costs of office space, utilities, insurance, and taxes.

Year	Quarter	App Purchases	Capex	Opex	Revenue	Break Even Indicator	Profit
2024	Q1	0	5,000	5,000	0	-10000	-10,000
2024	Q2	2,500	0	5,000	5,000	-10,000	0
2024	Q3	2,500	0	5,000	5,000	-10,000	0
2024	Q4	5,000	0	5,000	10,000	-5,000	5,000
2025	Q1	10,000	0	5,000	20,000	10,000	15,000
2025	Q2	15,000	0	5,000	30,000	35,000	25,000
2025	Q3	15,000	0	5,000	30,000	60,000	25,000
2025	Q4	20,000	0	5,000	40,000	95,000	35,000
2026	Q1	15,000	0	5,000	30,000	120,000	25,000
2026	Q2	17,500	0	5,000	35,000	150,000	30,000
2026	Q3	17,500	0	5,000	35,000	180,000	30,000
2026	Q4	20,000	0	5,000	40,000	215,000	35,000
2027	Q1	17,500	0	5,000	35,000	245,000	30,000
2027	Q2	17,500	0	5,000	35,000	275,000	30,000
2027	Q3	20,000	0	5,000	40,000	310,000	35,000
2027	Q4	25,000	0	5,000	50,000	355,000	45,000
2028	Q1	20,000	0	5,000	40,000	390,000	35,000
2028	Q2	25,000	0	5,000	50,000	435,000	45,000
2028	Q3	25,000	0	5,000	50,000	480,000	45,000
2028	Q4	30,000	0	5,000	60,000	535,000	55,000
2024-2028	total	320,000	5,000	100,000	640,000	_	535,000

Table 2.3 Projected revenue, costs, and profit

Cash Flow Analysis:

The table shows the projected revenue, costs, and profit for the project over the next 5 years, assuming that the project is priced at $2 per download.

- **Revenue**: The revenue is projected to increase steadily over the next 5 years, from $0 in Q1 2024 to $535,000 in Q4 2028. This is due to the increasing number of downloads.
- **Costs**: The costs are projected to remain relatively constant over the next 5 years, at $5,000 per quarter. This includes the costs of development, operations, and marketing.

- **Profit**: The profit is projected to be positive in Q1 2025 and continue to increase over the next 5 years, reaching **$535,000 in Q4 2028**. This is due to the increasing revenue and decreasing costs.
- The project will **break even** in **Q1 2025**, if the project can achieve the projected number of downloads and revenue.
- The total profit over the 5-year period is $535,000. This assumes that the project can achieve the projected number of downloads and revenue.
- The project is projected to 20,000 app purchases in Q1 2025. If the project can maintain this rate of growth, it will reach the break-even point in Q1 2025.

However, it is important to note that these are just projections, and the actual results may vary. The company should monitor the project's performance closely and adjust the pricing, marketing, and sales strategies as needed.

Chapter 3 Literature Survey

3.1 Background on Summarization

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 [9].

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 [10].

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:

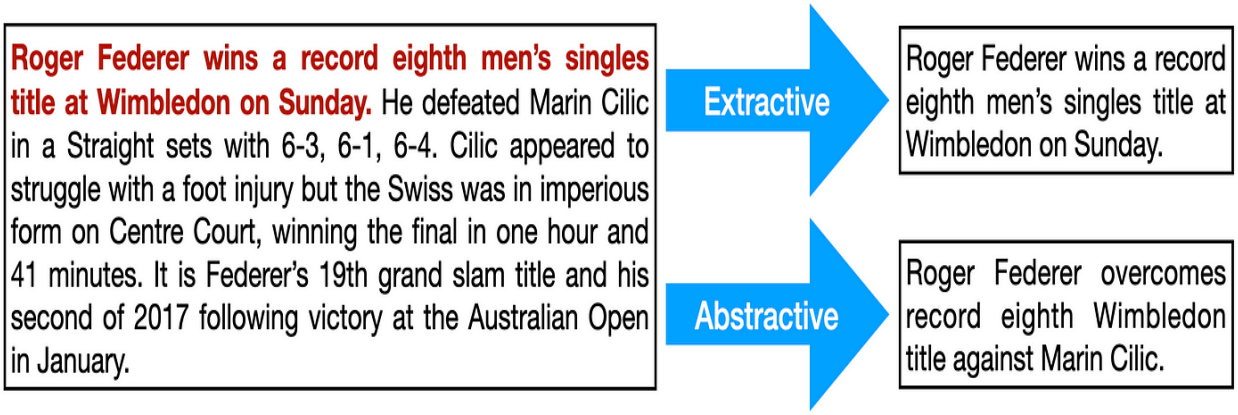


Figure 3.1 Extractive vs Abstractive Summarization Task.

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 [7]. 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.

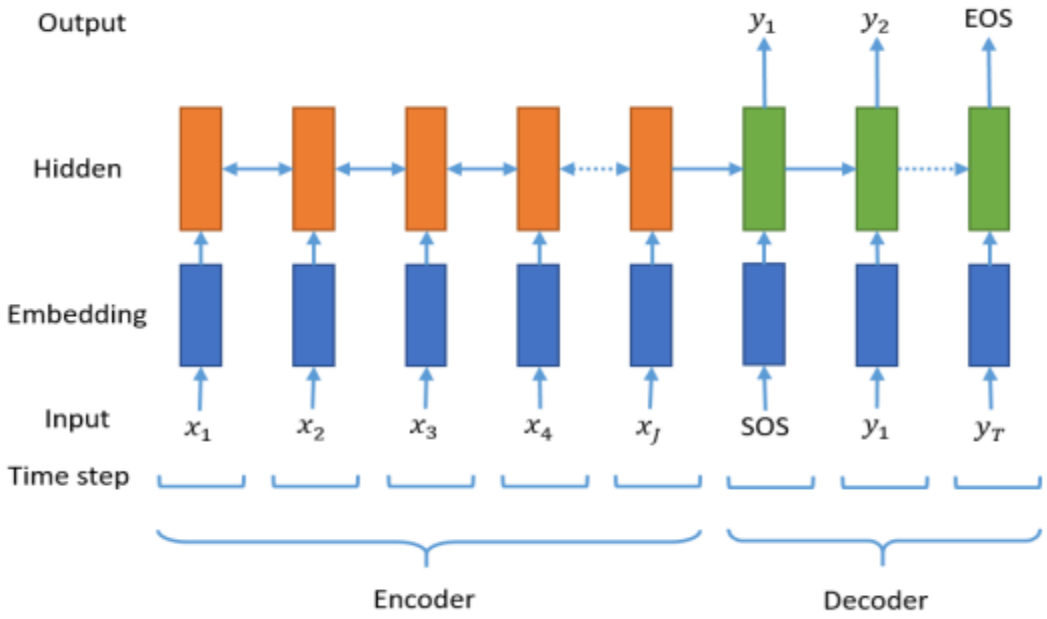


Figure 3.2 RNN for Abstractive Summarization.

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 [1].

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.

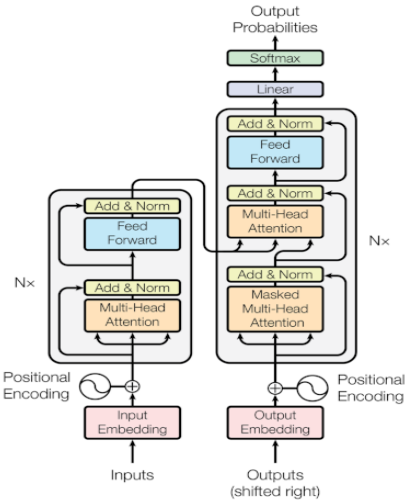


Figure 3.3 Encoder Decoder Transformer

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.

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 [1].

$$PE_{(pos,2i)} = \sin\left(pos/1000^{2i/d_{model}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(pos/1000^{2i/d_{model}}\right)$$

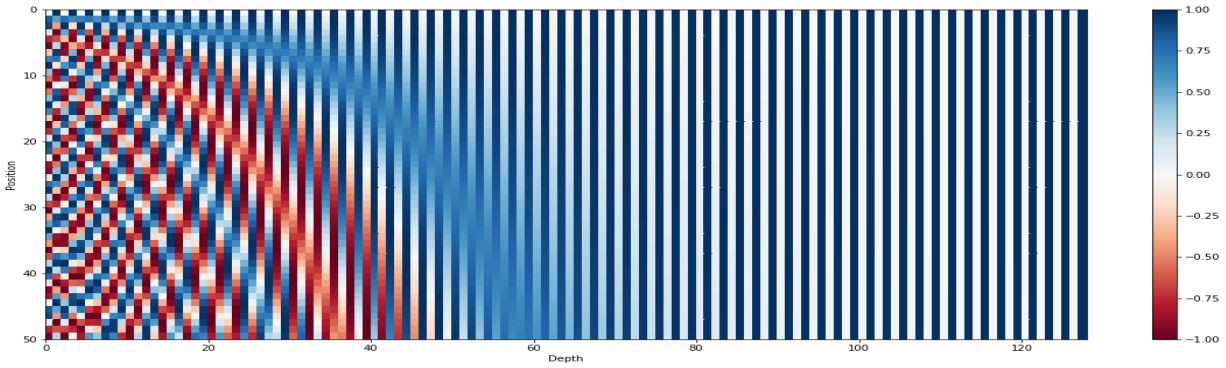


Figure 3.4 The 128-dimensional positional encoding for a sentence with the maximum length of 50. Each row represents the embedding vector.

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.

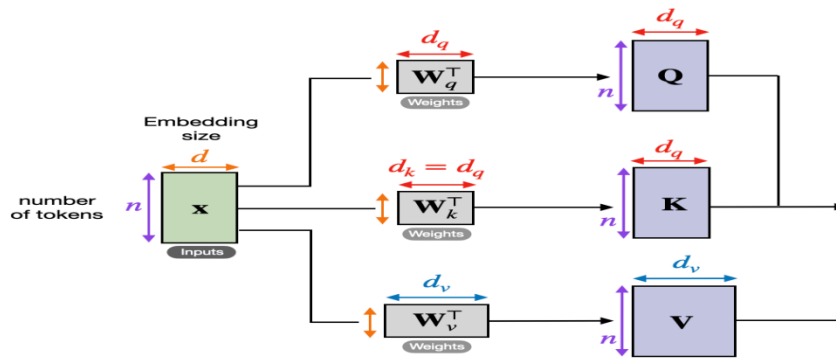


Figure 3.5 Self Attention Mechanism

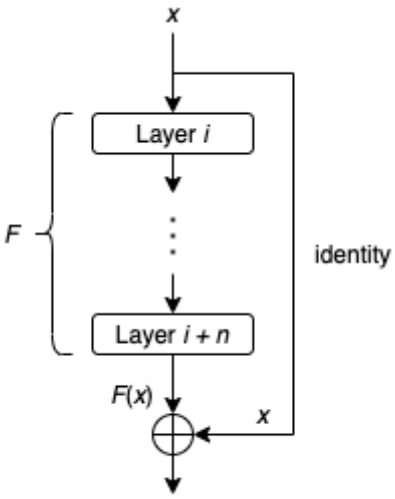


Figure 3.6 Residual Connections

3.1.2.1.2.2.1.3 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, elementwise. 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 [11].

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.

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.

$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2$$

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.

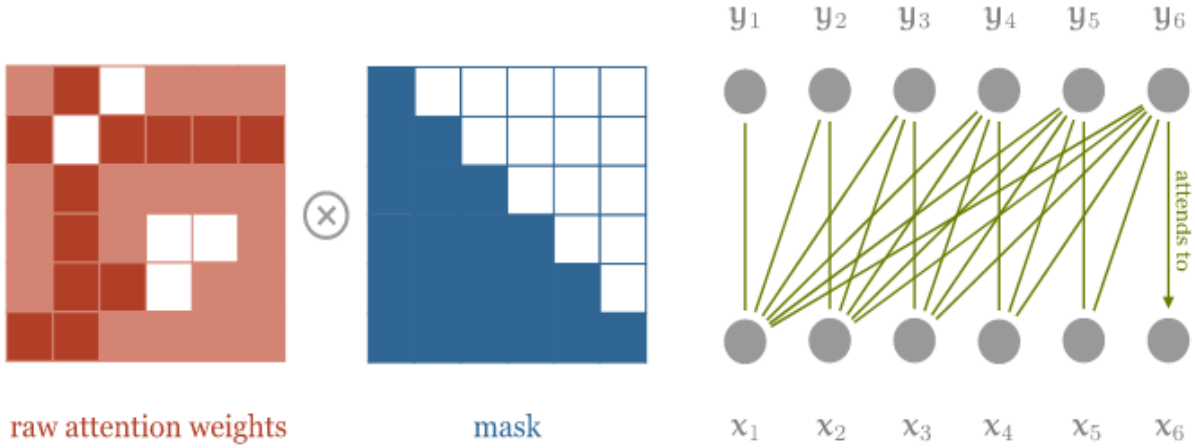


Figure 3.7 Masked Attention Mechanism

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.

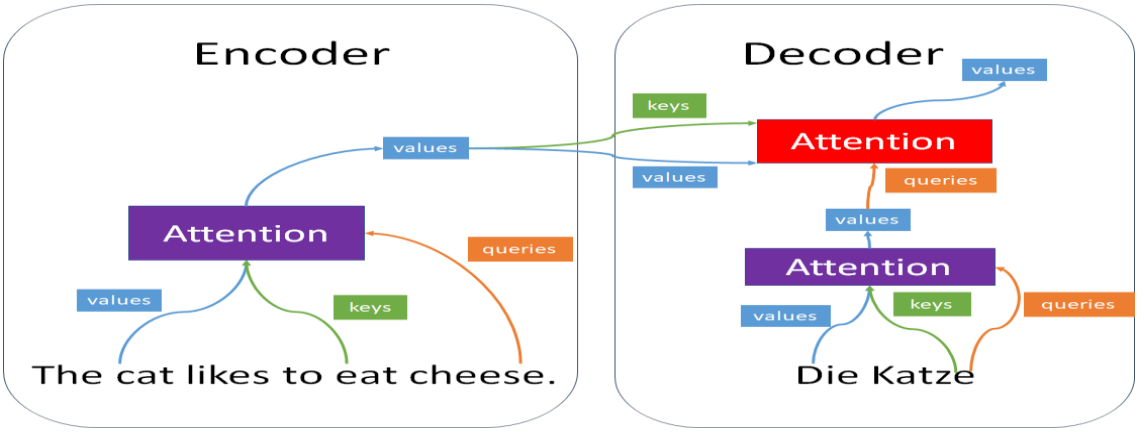


Figure 3.8 Cross attention Mechanism

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.

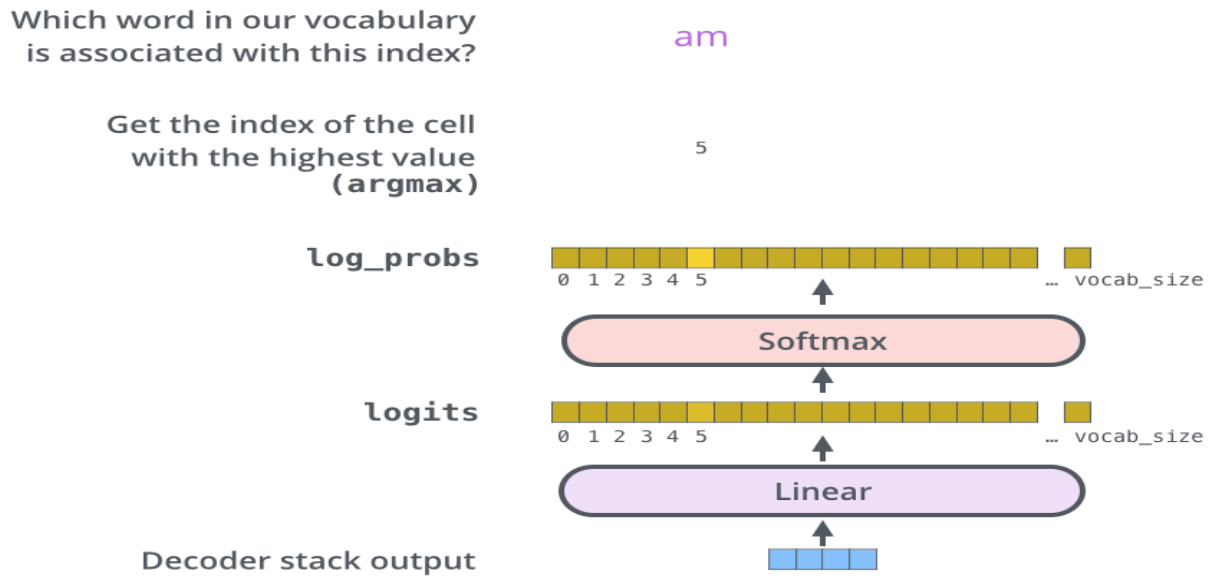


Figure 3.9 The Linear and Softmax Layers at the end

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 sub-word 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.

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.

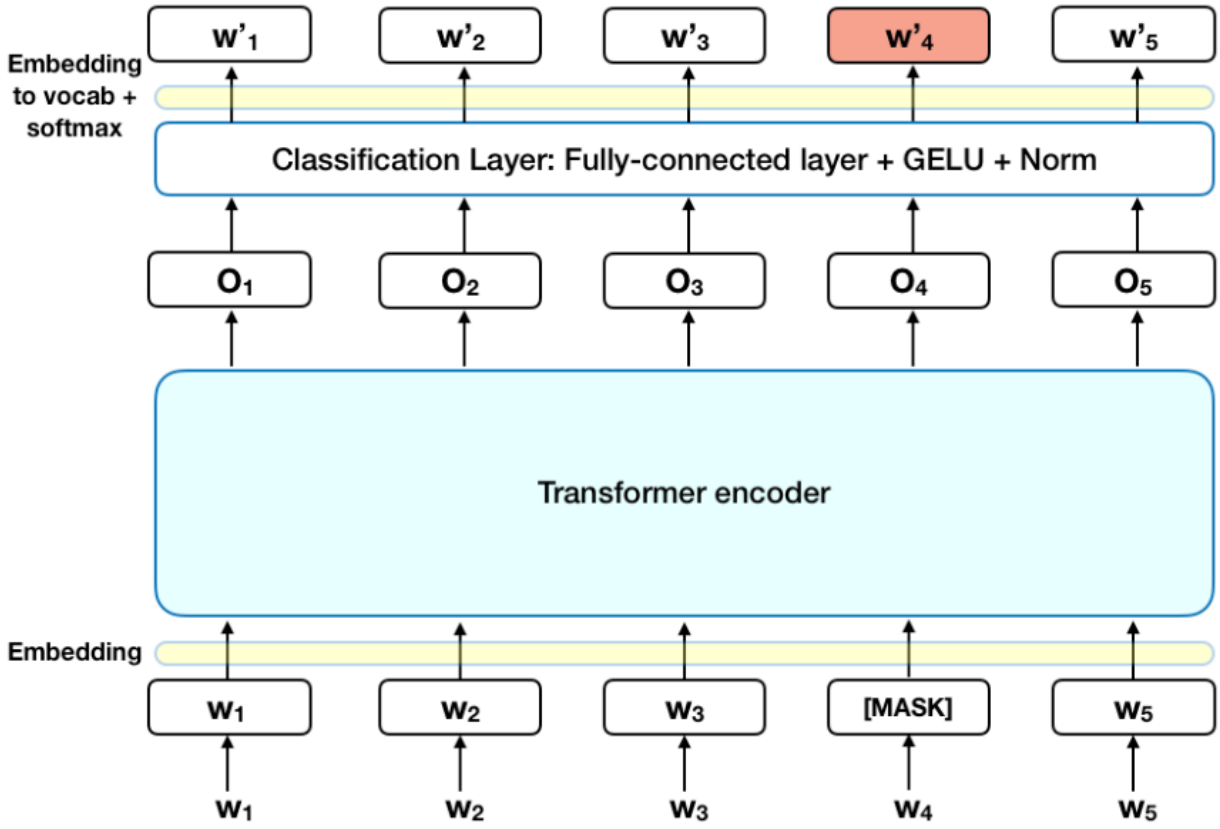


Figure 3.5 Encoder and Classifier on top

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 [8]. 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.

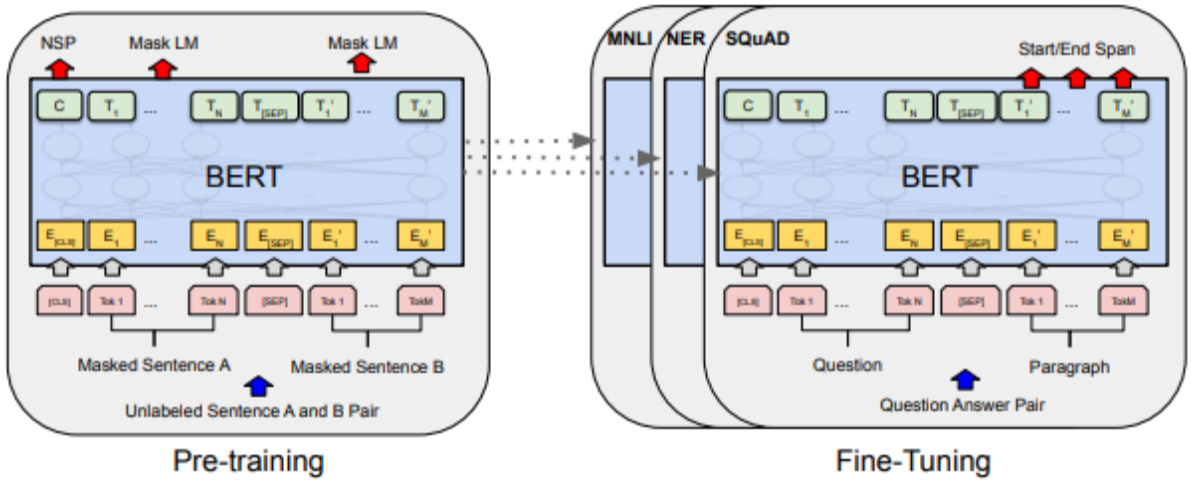


Figure 3.6 Overall pre-training and fine-tuning procedures for BERT

3.2 Background on Sentiment Analysis

3.2.1 non-Engineering background

3.2.1.1 Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a field within natural language processing (NLP) that focuses on understanding and extracting subjective information, such as sentiments, opinions, and emotions, from text data. It involves the use of computational methods and algorithms to automatically analyze and classify the sentiment expressed in written content, such as customer reviews, social media posts, or online comments.

One of the key applications of sentiment analysis is in understanding public opinion and customer feedback. By analyzing sentiments expressed in online reviews or social media posts, companies can gain valuable insights into customer satisfaction, identify emerging trends, and make data-driven decisions to improve their products or services.

Sentiment analysis also plays a crucial role in social media monitoring and brand reputation management. By monitoring sentiments expressed on platforms like Twitter, Facebook, or Instagram, businesses can gauge public sentiment towards their brand, track customer feedback, and proactively address any issues or concerns.

3.2.2 Engineering background

3.2.2.1 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.

$$\text{TF} = \frac{number\ of\ times\ the\ term\ appears\ in\ the\ document}{total\ number\ of\ terms\ in\ the\ document}$$

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.

$$\text{IDF} = \log \frac{number\ of\ documents\ in\ the\ corpus}{number\ of\ documents\ in\ the\ corpus\ containing\ the\ term}$$

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.

$$tf\_IDF = tf * IDF$$

3.2.2.2 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.

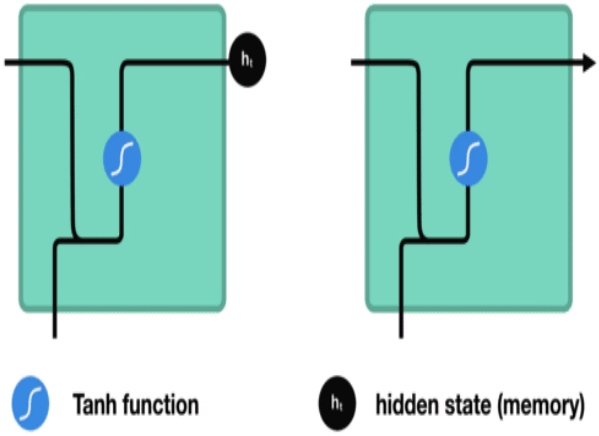


Figure 3.7 Flow of Hidden State through RNN.

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.

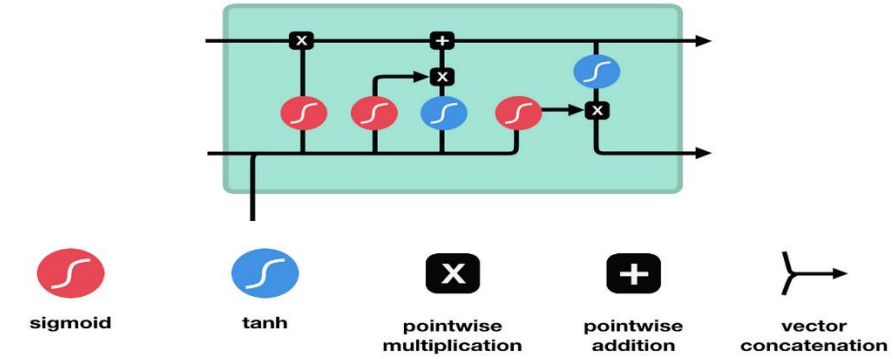


Figure 3.8 LSTM Cell

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.

3.2.2.3 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.

3.2.2.4 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 between 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:

$$J = \sum_{i,j=1}^{V} f(X_{ij})\left(w_i^T W_k + b_i + B_k - \log X_{ik}\right)^2$$

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.

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} & if\ x < x_{max} \\ 1 & otherwise \end{cases}$$

This expression ensures that f has values between 0 and 1, α 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.

3.2.2.5 SoftMax Activation function

The SoftMax function transforms the input values into a range between 0 and 1, ensuring that the sum of the transformed values is equal to 1. This property makes it suitable for representing probabilities across different classes.

The formula for the SoftMax activation function is as follows:

$$Softmax(x_i) = \frac{e^{x_i}}{sum(e^{x_i})\ \forall_j}$$

In the formula, x_i represents the input value for the i-th class, exp denotes the exponential function, and the sum is taken over all the classes j.

The SoftMax function is often used as the final activation function in the output layer of a neural network during multiclass classification tasks. It allows the network to produce a probability distribution over all possible classes, indicating the likelihood of the input belonging to each class.

By applying the SoftMax function, the output values represent the confidence scores or probabilities associated with each class. The class with the highest probability is typically chosen as the predicted class label.

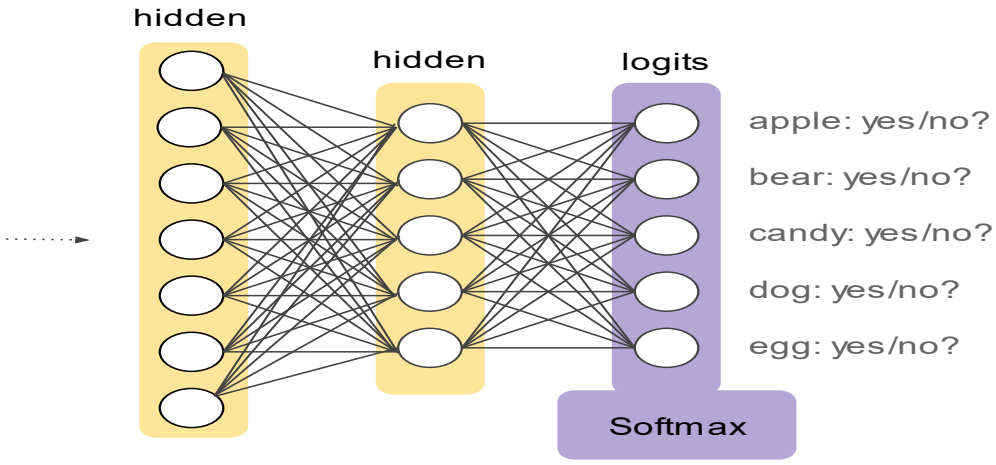


Figure 3.9 SoftMax activation function

The SoftMax activation function is differentiable, which is crucial for training neural networks using gradient-based optimization algorithms like backpropagation. The gradients of the SoftMax function are used to update the network's weights during the training process, enabling it to learn and improve its classification performance.

In summary, the SoftMax activation function is widely used in multiclass classification tasks to produce a probability distribution over classes. It transforms input values into a range between 0 and 1 and ensures that the transformed values sum up to 1. By providing confidence scores or probabilities for each class, the SoftMax function helps determine the most likely class label for a given input.

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 [16] 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.

These 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**.

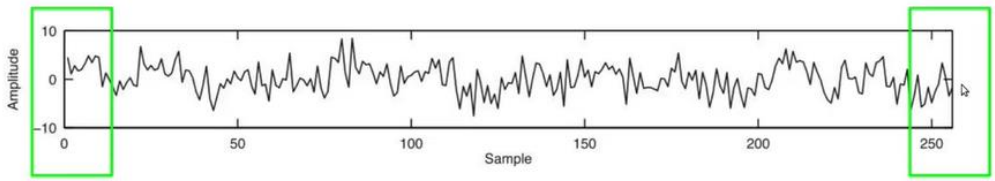


Figure 3.10 Windowing to overcome spectral leakage.

Windowing using Hann Window

$$w(k) = 0.5 \cdot (1 - \cos(\frac{2\pi k}{K-1})), k = 1...K$$

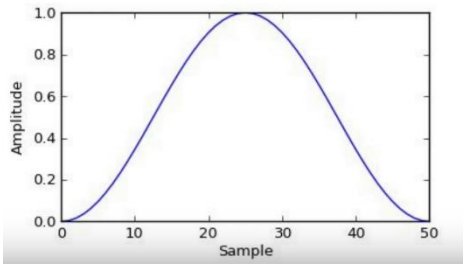


Figure 3.11 Hann window

Applying window fn. to signal formula:

Sw(k)= S(k) .w(k), k=1... K

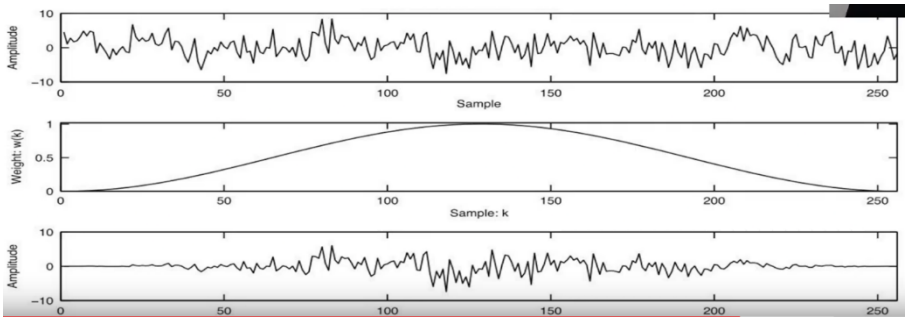


Figure 3.12 Applying Hann window on segment.

2.3.2 Short-Time Fourier Transform (STFT)

Why do we need STFT? What is wrong with normal Fourier transformation?[16]

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

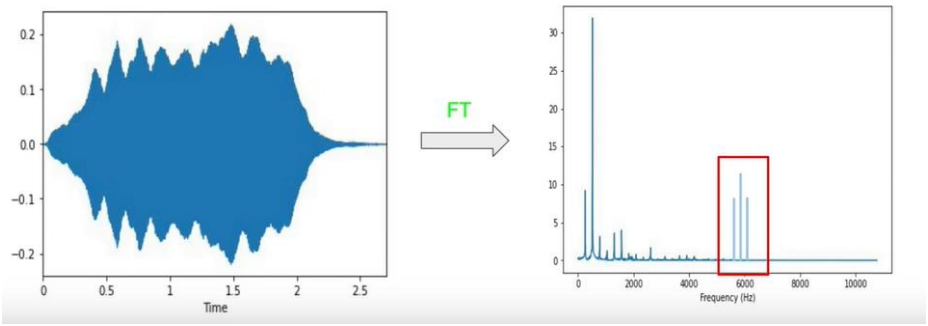


Figure 3.13 Fourier transform

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 [16]) 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.

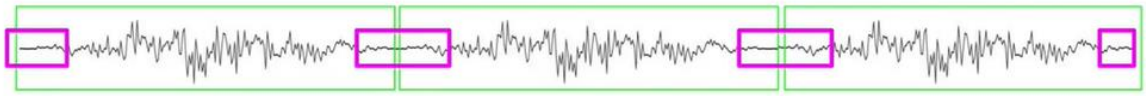


Figure 3.19 Information loss during framing

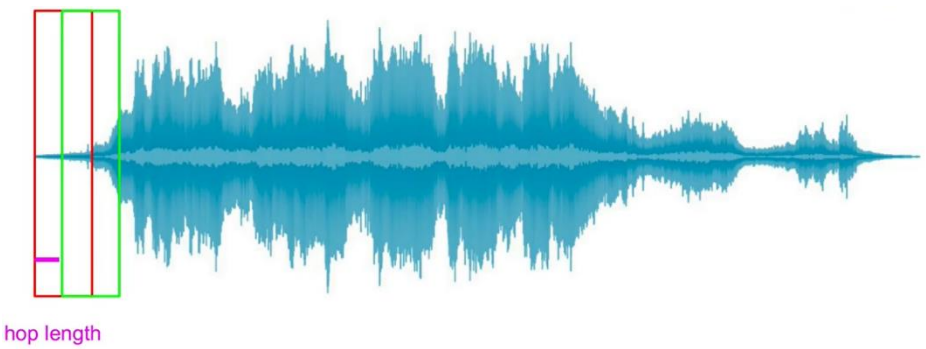


Figure 3.14 Overlapping frames

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.

The 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.

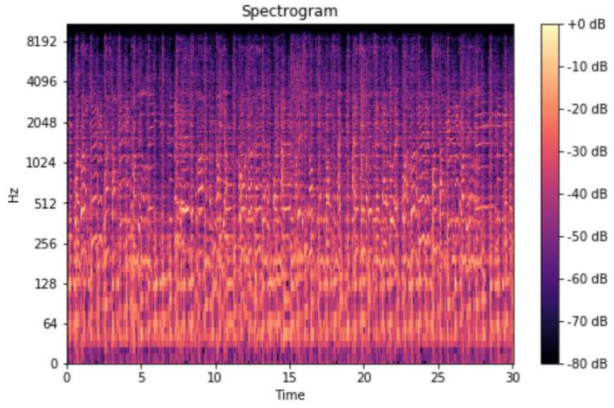


Figure 3.15 Mel spectogram

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.

2.3.5 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:

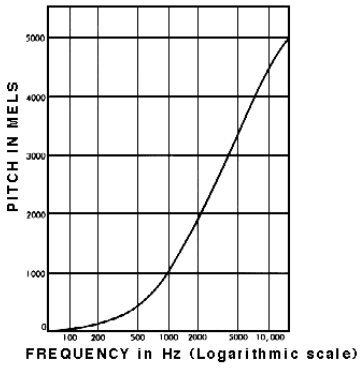


Figure 3.16 Mel Scale

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.

    a. 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)

b. 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.

3. 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.

4. 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.

5. 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

In recent years, avatar creation has gained attention 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 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.

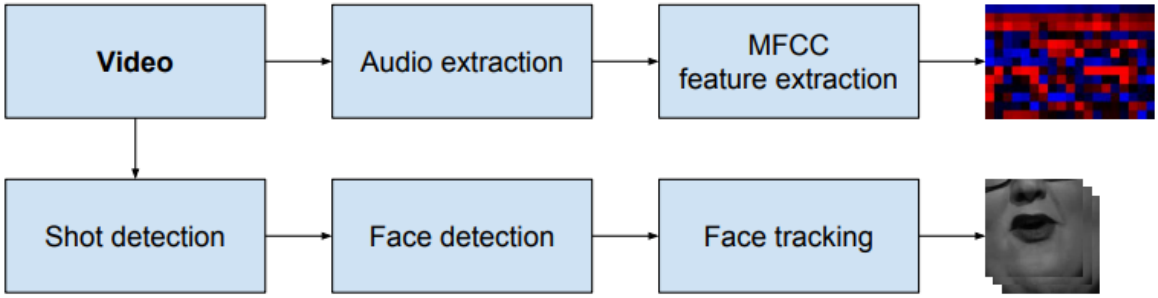


Fig. 4. Pipeline to generate the audio-visual dataset.

Figure 3.17 Wav2Lip Pipeline [21]

3.4.1 Face Encoder

The face encoder is CNN which is used to process the face sequence. It takes input frames of a video which are face images having three color channels (RGB). After that produces a feature vector representing the shape of the lip in the frame. CNN is trained on a dataset of video frames and their corresponding lip annotation which are created by manually tracking the movement of the lips in the video frames.

It is vital for the input face sequences' meaningful visual feature extraction. It captures hierarchical representations that encapsulate relevant face information by using a stack of convolutional layers. These characteristics are then used by the model's later components, such the binary predictor and lip-sync decoder, to produce precise lip motions that are synced with the input speech signal.

Figure 3.18 Face encoder architecture

3.4.2 Audio Encoder

The audio encoder is convolution neural network which is used to process an input speech signal and produce a feature vector representing the MFCCs of the speech. It is trained on a dataset of speech signals and corresponding MFCCs features. It uses signal processing techniques STFT, to extract spectrogram features from the audio waveform. By leveraging techniques such as STFT and spectrogram analysis, it extracts informative features that capture the speech characteristics. These features are then used in conjunction with the face encoder to generate accurate lip movements synchronized with the input audio.

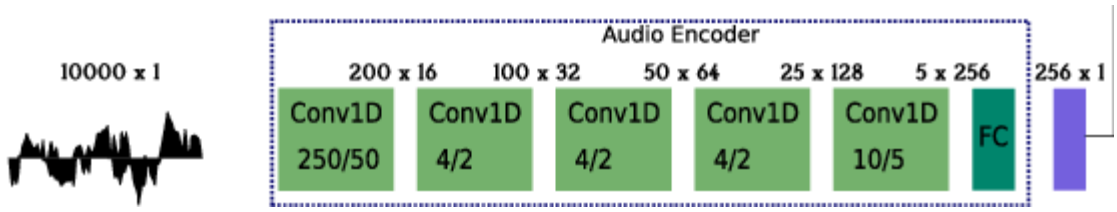


Figure 3.19 Audio encoder architecture

3.4.3 Face Decoder

The face decoder takes as input the feature vectors from the face encoder and the speech encoder and outputs a sequence of control points that define the shape of the lips in the output video. The decoder is a CNN that is trained on a dataset of video frames and corresponding control points. The control points are typically generated by manually animating the lips in the video frames. By combining the audio and visual cues, it generates realistic lip movements and facial expressions that are synchronized with the input audio, resulting in a convincing lip-syncing effect in the final synthesized face images.

3.4.4 Lip Sync Discriminator

The aim of the lip sync discriminator is to distinguish between fake and real lip movements. It is used to improve the realism of the generated lip movements. It learns to identify the inconsistencies and artifacts in the synthesized face images and their synchronization with the audio. The discriminator assigns a probability score to the input lip-synced video, indicating the likelihood of it being real or fake. It is trained on real videos from the dataset, and the fake videos are from the generated video from our model.

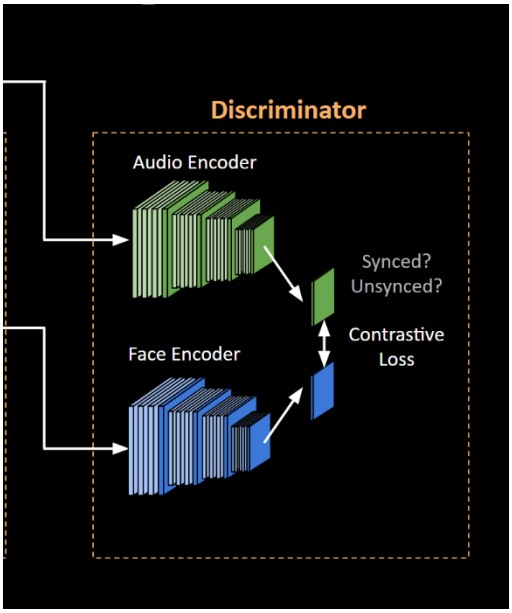


Figure 3.20 Lip Discriminator [22]

The Lip Sync Discriminator is typically a CNN which is used for the lip sync detection. It consists of multiple convolutional layers followed by activation functions such as LeakyReLU.

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:

Sentiment analysis is a crucial component in natural language processing (NLP) that focuses on extracting and understanding subjective information from text data. It plays a vital role in various domains, such as customer feedback analysis, social media monitoring, and market research. In recent years, significant advancements have been made in sentiment analysis techniques, employing different approaches and models.

Some of the used approaches and recent publications that have been utilized for sentiment analysis:

1. Recurrent Neural Networks (RNNs):
   RNNs are a class of neural networks that excel at processing sequential data, making them well-suited for sentiment analysis tasks where the order of words in a sentence matters. Traditional RNNs suffer from the vanishing gradient problem, which hampers their ability to capture long-term dependencies.

   a. "Aspect-Level Sentiment Classification with Recurrent Neural Networks" by Tang et al. (2017): This study proposes a recurrent neural network (RNN) model for aspect-level sentiment classification. The model employs RNNs to capture contextual information and utilizes attention mechanisms to focus on important aspects within the input text.

   b. "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification" by Yang et al. (2018): This paper introduces a gated recurrent neural network (GRNN) model for sentiment classification. The model incorporates a document-level gate mechanism that captures different aspects of sentiment expressed in a document, improving overall sentiment analysis performance.

2. Long Short-Term Memory (LSTMS)
   LSTM networks are a type of recurrent neural network (RNN) that have shown effectiveness in capturing long-term dependencies in sequential data, making them suitable for sentiment analysis tasks. LSTMs address the vanishing gradient problem faced by traditional RNNs, enabling them to retain and utilize information over longer sequences.

   a. "Bidirectional LSTM-CRF Models for Sentiment Analysis" by Ma et al. (2017): This study proposes bidirectional LSTM model for sentiment analysis. By incorporating bidirectional LSTMs), the model captures contextual information and model label dependencies, respectively. The combination of these techniques leads to enhanced sentiment classification accuracy.

   b. "A Hierarchical Attention Model for Sentiment Analysis" by Yang et al. (2018): This paper proposes a hierarchical attention model that employs LSTMs to capture both the word and sentence-level representations in sentiment analysis. The model incorporates attention mechanisms to focus on important words and sentences for accurate sentiment classification.

3. Transformers: Transformers have gained significant attention in recent years due to their ability to capture long-range dependencies in text data effectively. Transformer-based models, such as BERT (Bidirectional Encoder Representations from Transformers), have shown remarkable performance in sentiment analysis tasks by learning contextualized word representations.

    a. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by Devlin et al. (2018): This influential paper introduces BERT, a pretraining model that utilizes transformer-based architectures. BERT achieves state-of-the-art results on various NLP tasks, including sentiment analysis, by training on large-scale unlabeled data and then fine-tuning on specific tasks.

    b. "RoBERTa: A Robustly Optimized BERT Pretraining Approach" by Liu et al. (2019): This work presents RoBERTa, an optimized variant of BERT. It employs larger scale training data and longer training duration to improve BERT's performance. RoBERTa outperforms BERT on several sentiment analysis benchmarks, showcasing the effectiveness of transformer-based models.

3.5.4 Genre Classification:

The 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:

There are several differences between our work and the previous work. The Lip Sync Discriminator has the major changes.

1. The input image was grey scale images, so we changed it to the original image.

2. The model is significantly deeper, with residual skip connections.

3. Previously the loss function was Euclidean distance between the network outputs is minimized or maximized it was used in Towards Automatic Face-to-Face Translation paper. The new function is cosine-similarity with binary cross-entropy loss. This function consists of a dot product between the ReLU-activated video and speech embeddings v, s to yield a single value between [0, 1] for each sample that indicates the probability that the input audio-video pair is in sync.

1. Old function [23]

   a) $E = \frac{1}{2N} \sum_{n=1}^{N} (y_n) d_n^2 + (1 - y_n) \max(margin - d_n, 0)^2$

   b) $d_n = \left|| v_n - a_n \right||_2$

2. New function [20]

   a) $P_{sync} = \frac{v.s}{\max(||v||_2 . ||s||_2, \in)}$

1. A trained lip sync discriminator was used. The accuracy of the old one was 56%, our used lip sync discriminator's accuracy is 91% accuracy. The higher the accuracy means that the discriminator can differentiate between real and fake lip synchronization which is useful during the training. This helps in assessing the quality of the lip-synced videos.

2. Rigorous evaluation benchmarks and metrics to accurately measure lip synchronization in unconstrained videos. Extensive quantitative evaluations on our challenging benchmarks show that the lip-sync accuracy of the videos generated by our Wav2Lip model is almost as good as real synced videos. The used evaluation benchmarks: [20]

   a. LSE-D: It is a metric used to measure the quality of lip sync in a generated video. It is calculated as the Euclidean distance between the predicted and ground truth lip positions. A lower LSE-D value indicates a higher quality lip sync.

   b. LSE-C: It is a metric used to measure the quality of lip sync in a generated video. It is calculated as the mean squared error between the predicted and ground truth lip positions. A higher LSE-C value indicates a lower quality lip sync.

   c. FID: It is a metric used to assess the quality of images created by a generative model

3.6 Implemented Approach

3.6.1 Summarization

The implemented approach in this study involves two models for text summarization: an extractive summarizer based on BERT and an abstractive summarizer built from scratch. The first model utilizes the powerful BERT (Bidirectional Encoder Representations from Transformers) framework. It employs a pre-trained BERT model for sentence encoding and representation learning. By leveraging BERT's contextual understanding of language, the extractive model can extract salient sentences directly from the source document to generate the summary. This extractive approach aims to capture important

information without introducing additional content. The second model focuses on abstractive summarization and is developed from scratch. Unlike extractive methods that select and rearrange sentences, the abstractive summarizer generates a summary by comprehending the source document and generating novel sentences that capture the essential information. This model employs a combination of natural language processing techniques, deep learning architectures, and text generation algorithms to produce coherent and concise summaries. By implementing both extractive and abstractive summarization approaches, this study explores the strengths and limitations of each technique. The extractive model harnesses the power of pre-trained language models and sentence encoding, while the abstractive model leverages the flexibility of generating new sentences. Comparing the results and evaluating the performance of these two models provides valuable insights into their effectiveness and suitability for different summarization tasks.

3.6.2 Text to Speech

This section provides an overview of the implemented approach in the SpeakNews project for text-to-speech functionality. The Tacotron model [13], a prominent and effective method in the field of speech synthesis, was chosen as the foundation for generating human-like speech from text. This paper delves into the technical details and implementation aspects of the Tacotron model, highlighting its architecture, training process, and performance evaluation. The utilization of Tacotron in the SpeakNews app has played a crucial role in enhancing the user experience and delivering high-quality audio content to the target audience.

Introduction:

Text-to-speech (TTS) technology has made significant advancements in recent years, enabling machines to convert written text into natural-sounding speech. The implementation of TTS in the SpeakNews project aims to transform news summaries into spoken audio, providing users with an engaging and accessible news consumption experience. Among various TTS models, Tacotron has emerged as a state-of-the-art approach, renowned for its ability to generate expressive and human-like speech.

Tacotron Model Architecture:

The Tacotron model [13] is based on a sequence-to-sequence architecture with an attention mechanism. It consists of two primary components: the text encoder and the decoder. The text encoder processes the input text and extracts high-level linguistic features. These features are then passed to the attention mechanism, which focuses on relevant parts of the text during the synthesis process. The decoder utilizes the attention mechanism and the encoded linguistic features to generate mel-spectrogram frames, which are finally converted into waveform audio using a vocoder.

Training Process: Training the Tacotron model involves two stages: teacher forcing and inference. In the teacher forcing stage, the model is trained using aligned pairs of text and corresponding mel-spectrograms. The objective is to minimize the difference between the

generated mel-spectrograms and the ground truth spectrograms. In the inference stage, the trained model is used to generate mel-spectrograms for unseen text inputs.

To facilitate training, a large data set comprising text and corresponding speech recordings is required. This dataset is typically obtained by collecting and aligning text and speech pairs from various sources. The Tacotron model is trained using LJ Speech dataset through an iterative process, optimizing the model's parameters to generate accurate and natural-sounding speech.

Performance Evaluation: The performance of the Tacotron model is evaluated using objective and subjective measures. Objective measures include metrics such as mel-spectrogram similarity, alignment accuracy, and pitch contour analysis. Subjective evaluation involves conducting listening tests where human evaluators rate the quality, naturalness, and intelligibility of synthesized speech. These evaluations help assess the model's ability to produce high-quality speech output.

Integration in SpeakNews: The Tacotron model serves as the backbone of the text-to-speech functionality in the SpeakNews app. By leveraging Tacotron's capabilities, the app delivers news summaries in a compelling and human-like audio format. Users benefit from the clear and natural-sounding speech output, which enhances the news consumption experience and provides a seamless transition from text to spoken content.

The implementation of the Tacotron model in the SpeakNews project has demonstrated its effectiveness in generating high-quality speech from text inputs. The model's architecture, training process, and performance evaluation contribute to its success in delivering expressive and natural-sounding audio content. By incorporating Tacotron into SpeakNews, users can experience a captivating and engaging news consumption experience through the app's text-to-speech functionality. Continued research and advancements in TTS models like Tacotron will further improve the quality and realism of synthesized speech, paving the way for enhanced user experiences in the future.

3.6.3 Sentiment Analysis

In our sentiment analysis project, we implemented an approach based on Long Short-Term Memory (LSTM) networks, which we found to be highly effective for capturing sequential dependencies and context in text data. Our choice of LSTMs was driven by several advantages that distinguish them from other approaches in sentiment analysis.

In our comprehensive evaluation of various approaches for sentiment analysis, we conducted extensive experiments to assess their performance and effectiveness. After considering and comparing multiple approaches, including rule-based methods, traditional machine learning algorithms, and other deep learning models, we found that our implemented LSTM approach outperformed all other approaches in terms of sentiment analysis accuracy and overall performance.

Firstly, LSTMs excel at modeling long-term dependencies, which is essential for sentiment analysis tasks where the sentiment expressed in a sentence or document can

be influenced by words or phrases appearing far apart. The ability of LSTMs to retain and utilize information over longer sequences makes them well-suited for capturing nuanced sentiment information.

Secondly, the bidirectional nature of LSTM networks allows them to consider information from both past and future contexts. This bidirectional modeling enables a more comprehensive understanding of sentiment by incorporating contextual cues from both preceding and succeeding words. By capturing dependencies in both directions, our LSTM-based approach can effectively capture the overall sentiment expressed in the text.

Moreover, LSTMs are capable of learning and representing complex patterns in the data. They can capture both local and global contextual information, allowing for a deeper understanding of sentiment-related features. This enables our approach to capture sentiment nuances and variations, leading to improved sentiment classification accuracy.

Furthermore, our LSTM-based approach benefits from attention mechanisms, which enable the model to selectively focus on relevant words or phrases within the input sequence. By attending to important sentiment-related information, our approach can effectively highlight and prioritize the most influential parts of the text for sentiment classification.

3.6.4 Avatar Generation

The implemented approach of our mode involved a comprehensive model architecture. The architecture consists of a face encoder, audio encoder, face decoder, and lip sync discriminator. First, the face encoder extracted facial features from input face images through a series of convolutional layers that progressively down sampled the images. Second, the audio encoder processed the input audio sequences, extracting audio features that were relevant to lip movements. Third, the face decoder combined the extracted facial and audio features by employing transposed convolutional layers to generate synchronized and realistic lip movements. Finally, we assessed the quality of the generated lip-synced videos, the lip sync discriminator was employed which. consisted of convolutional layers, evaluated the realism of the lip movements by distinguishing between real and synthesized videos. The training process involved optimizing the model using various loss functions such as reconstruction loss, adversarial loss, and perceptual loss. These losses ensured that the generated lip movements closely matched the ground truth and were indistinguishable from real lip movements. Evaluation metrics, including lip sync discriminator accuracy, were used to quantify the model's performance. The lip sync discriminator accuracy represented the percentage of correctly classified lip-synced videos as real or fake, indicating the level of realism achieved by the model.

Chapter 4 System Design and Architecture

4.1 Overview and Assumptions

The project aims to develop a news summarizer that provides users with two options for consuming news content. The first option is to listen to an audio summary, while the second option is to watch a video of an avatar delivering the news. The summarizer goes through several stages to process the input text and generate the desired outputs.

1. Tokenization and Summarization: The initial stage involves tokenizing the input text and employing two types of summarizers: extractive and abstractive. The extractive summarizer utilizes an encoder-only transformer, followed by a classifier to determine which sentences should be included in the summary. The abstractive summarizer, on the other hand, uses an encode-decode transformer to generate a summary.

2. Sentiment Analysis and News Genre Classification: The output of the summarizer is then fed into three subsequent modules. The first module is sentiment analysis, where the text passes through embedding layers, followed by three LSTM layers and a dense layer with SoftMax activation. This analysis aims to detect the sentiment expressed in the news content. The second module is news genre classification, which categorizes the news into specific genres. Similar to sentiment analysis, the text undergoes embedding, LSTM, and dense layers to classify the genre.

3. Text-to-Speech Conversion: The third module focuses on converting the generated summary into an audio format. The summary text is processed, and then a mel spectrogram is generated from it. Finally, the Griffin-Lim algorithm is applied to produce a WAV file, resulting in synthesized speech.

4. Avatar Generation: In the avatar generation stage, the input image or video is passed through a face encoder, while the input audio is also processed. The output from both encoders is then fed into a decoder, which combines audio and visual cues to generate realistic lip movements and facial expressions. The final synthesized face images exhibit lip-syncing effects that are synchronized with the input audio, creating a convincing avatar delivering the news.

Assumptions:

1. The project assumes the availability of a large dataset for training the various models involved, including the summarizers, sentiment analysis, news genre classification, and avatar generation.

2. It assumes that the encoder-only transformer and encode-decode transformer models have been trained on appropriate text summarization datasets.

3. The sentiment analysis model assumes the availability of a labeled sentiment dataset for training the embedding layers, LSTM layers, and dense layer.

4. The news genre classification model assumes access to a labeled dataset for training the embedding layers, LSTM layers, and dense layer to classify news articles into specific genres.

5. The text-to-speech module assumes the presence of a pre-trained model capable of converting text into mel spectrograms, as well as the availability of a dataset for training the Griffin-Lim algorithm.

6. The avatar generation module assumes the availability of pre-trained models for face encoding, lip-syncing, and facial expression generation, as well as relevant datasets for training these models.

4.2 System Architecture

4.2.1 Block Diagram

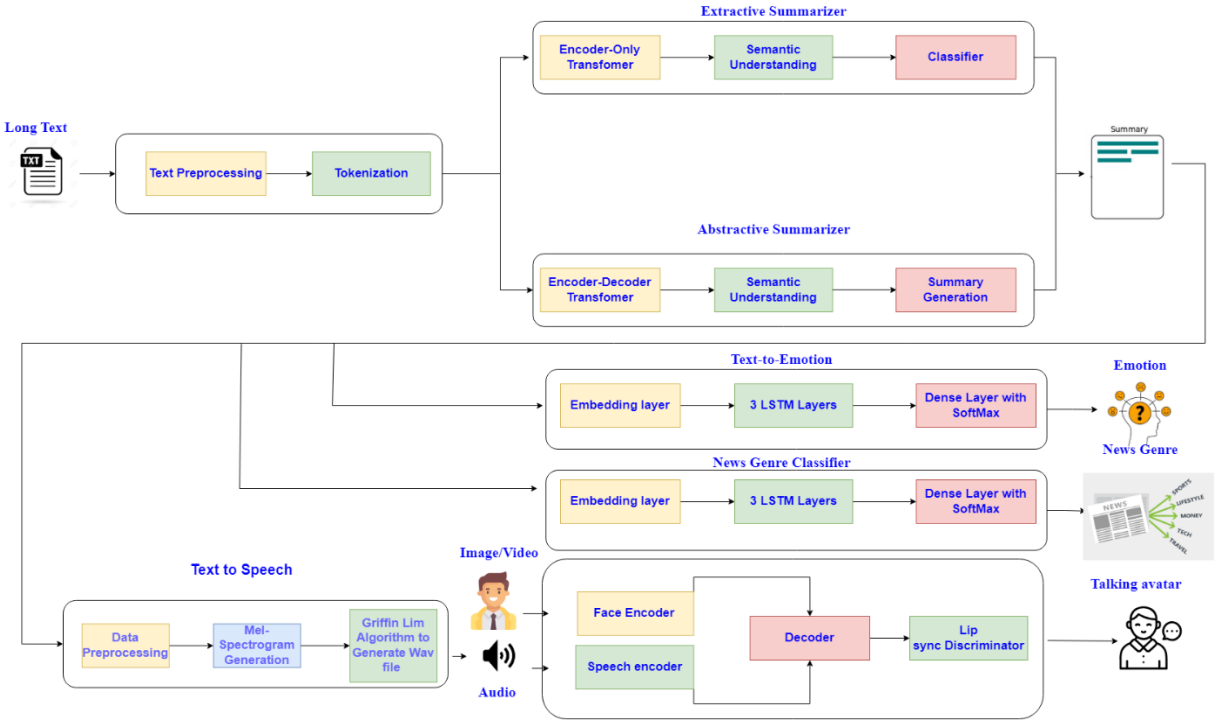


Figure 4.1 Full system architecture

4.3 Extractive Summarizer

The extractive summarization module plays a crucial role in applications where preserving the original wording and structure of the text is essential, such as news summarization, document summarization, and information retrieval systems. By effectively condensing lengthy documents into concise summaries, extractive summarizers enable users to quickly grasp the main ideas and essential information contained within a text, saving time and enhancing information accessibility.

4.3.1 Functional Description

4.3.1.1 Input

In the context of extractive summarization, the input to the summarizer is a source document that contains multiple sentences or phrases. This source document can be a lengthy article, a news report, a research paper, or any other text that requires summarization. The input serves as the basis for extracting key information and identifying important sentences or phrases that capture the essence of the document.

4.3.1.2 Output

The output is a condensed version of the source document consisting of selected important sentences or phrases.

4.3.1.3 Functionality

1. Preprocessing:

    - Input: Source document
    - Output: Preprocessed sentences or phrases
    - Function: Tokenizes the source document, removes stop words and punctuation, and applies text normalization techniques like stemming or lemmatization.

2. Encoding:

    - Input: Preprocessed sentences or phrases
    - Output: Encoded representations
    - Function: Uses an encoder only transformer to capture contextual information and semantic meaning of the input text. Generates high-dimensional representations for each sentence or phrase.

3. Sentence Scoring:

    - Input: Encoded representations
    - Output: Scores for each sentence or phrase
    - Function: Applies a scoring mechanism considering factors such as sentence length, keyword relevance, positional importance, or semantic similarity. Assigns a weight or score to each sentence or phrase indicating its importance for the summary.

4. Sentence Selection:

    - Input: Scores for each sentence or phrase
    - Output: Selected sentences or phrases

- Function: Selects sentences or phrases with the highest scores, filtering out those below a certain threshold. Arranges the selected sentences or phrases in the desired order to form the summary.

5. Post-processing:

    - Input: Selected sentences or phrases
    - Output: Final summary
    - Function: Performs post-processing steps to enhance readability, coherence, or grammatical correctness. This may include sentence compression, paraphrasing, or linguistic adjustments.

4.3.2 Modular Decomposition

1. Tokenization: The first step in the encoder is tokenization, where the input document is split into individual tokens or subwords. This is done using a tokenizer specific to BERT, which handles issues such as out-of-vocabulary words and word splitting. The tokenizer maps each token to its corresponding index in the vocabulary.

2. Input Embedding: Once the document is tokenized, the tokens are converted into dense vector representations called embeddings. These embeddings capture the semantic and contextual information of each token. In BERT, the input embeddings are learned during the pre-training phase. The input embedding layer maps each token index to its corresponding embedding vector.

3. Positional Encoding: To preserve the positional information of the tokens in the document, BERT incorporates positional encoding. Positional encoding adds a fixed-length vector to each token's embedding, indicating its position in the sequence. This helps the model differentiate between tokens based on their relative positions and capture the sequential dependencies in the document.

4. Transformer Encoder Layers: The core component of the encoder is the stack of transformer encoder layers. Each encoder layer consists of two sub-layers: the multi-head self-attention mechanism and the position-wise feed-forward neural network.

    - Multi-Head Self-Attention: The self-attention mechanism allows the model to attend to different parts of the input document while capturing the relationships between tokens. It computes the importance of each token in the context of the entire document. BERT utilizes multi-head self-attention, which performs self-attention computation multiple times with different learned linear projections, allowing the model to focus on different aspects of the input.
    - Position-Wise Feed-Forward Network: After the self-attention sub-layer, a position-wise feed-forward neural network is applied to each token independently. This network consists of two linear transformations with a ReLU

activation function in between. It helps to capture complex non-linear interactions between tokens and further refine the representations.

5. Layer Normalization: Layer normalization is applied after each sub-layer in the transformer encoder. It normalizes the values of the tokens within each layer, ensuring that the model can learn stable and effective representations. It helps in reducing the internal covariate shift and enables faster training convergence.

4.3.3 Design Constraints

One important design constraint is the length constraint, which restricts the summary to a specified length limit. This constraint ensures that the summary remains concise and captures the essential information within a given length. By adhering to the length constraint, the summarizer produces summaries that are focused and avoid unnecessary details [6].

4.3.4 Input dataset

The input dataset for the extractive summarizer using the CNN/Daily Mail dataset consists of a large collection of news articles paired with corresponding human-generated summaries. The CNN/Daily Mail dataset is a popular benchmark dataset commonly used in the field of extractive summarization. It includes news articles from the CNN and Daily Mail websites, along with multiple highlight sentences that serve as summaries of the articles.

4.3.4.1 Preparing the dataset for the Extractive Summarization Task

The process of preparing the CNN/Daily Mail dataset for the extractive summarization task involves several steps, each serving a specific purpose in extracting important sentences from the source document.

1. **ROUGE Score Calculation**: The first step is to calculate the ROUGE scores, which are used to evaluate the quality of the generated summary. ROUGE scores measure the overlap between the extracted summary sentences and the reference (gold standard) summary sentences. By comparing the n-grams (1-grams and 2-grams) in the evaluated summary with those in the reference summary, the ROUGE scores provide an assessment of how well the extractive summary captures the content of the original document.

2. **Greedy Selection Algorithm**: The next step involves implementing a greedy selection algorithm to choose the most informative sentences from the document summary based on their ROUGE scores. The algorithm aims to maximize the ROUGE score by iteratively selecting sentences that contribute the most to the overall score. The selected sentences are added to a list and their indices are tracked.

3. **Sentence Cleaning**: Before calculating the ROUGE scores, the sentences in the document summary and the reference summary are cleaned by removing non-alphanumeric characters. This ensures that the comparison between the evaluated and reference n-grams is based on meaningful and consistent tokens.

The logic behind this approach is to leverage the ROUGE scores as a measure of the relevance and importance of sentences in the document summary. By selecting sentences with higher ROUGE scores, we prioritize those that have a stronger overlap with the reference summary, indicating their importance in capturing the essence of the original document. The greedy selection algorithm iteratively adds sentences to the summary based on their ROUGE scores, gradually constructing a concise summary that represents the key information from the source document.

Overall, this process combines the calculation of ROUGE scores, the application of a greedy selection algorithm, and the cleaning of sentences to extract the most relevant and informative sentences from the CNN/Daily Mail dataset for the extractive summarization task.

4.3.5 Tokenization and word embedding

4.3.5.1 Tokenization

Tokenization involves breaking down the input text into a sequence of subword tokens. Unlike traditional tokenization, which splits text at the word level, our model tokenizes text into subword units using a technique called WordPiece tokenization. This process allows us to handle out-of-vocabulary words and capture more fine-grained information.

During tokenization, special tokens such as [CLS] and [SEP] are added to mark the beginning and separation of sentences or documents. The input text is then split into fixed-length segments or truncated to fit the model's input length limit.

4.3.5.2 Word Embedding

Word Embedding: Word embedding in the extractive summarization module is achieved using pre-trained BERT (Bidirectional Encoder Representations from Transformers) models. BERT models are trained on large amounts of text data and learn contextualized representations of words.

4.3.6 BERT

4.3.6.1 Original BERT

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art language representation model introduced by Google in 2018. It revolutionized the field of natural language processing by introducing a pre-trained model that captures deep contextualized representations of words and sentences. BERT is trained on large

amounts of unlabeled text data from the internet, allowing it to learn rich and meaningful language representations.

The core idea behind BERT is to leverage a transformer architecture, which is a type of deep neural network specifically designed for sequence modeling tasks. Transformers are based on the concept of self-attention, allowing the model to attend to different parts of the input sequence and capture the dependencies between words or tokens.

BERT takes advantage of the bidirectional nature of transformers by considering both the left and right context of each word during training. This bidirectional modeling enables BERT to understand the context and meaning of a word based on the entire sentence or paragraph, rather than relying solely on left-to-right or right-to-left context.

The BERT model consists of several layers, typically with 12 or 24 transformer blocks. Each transformer block contains a multi-head self-attention mechanism and a feed-forward neural network. The self-attention mechanism allows BERT to capture the relationships between words in a sentence by assigning attention weights to different positions. This enables the model to focus on the most relevant words and create contextualized representations.

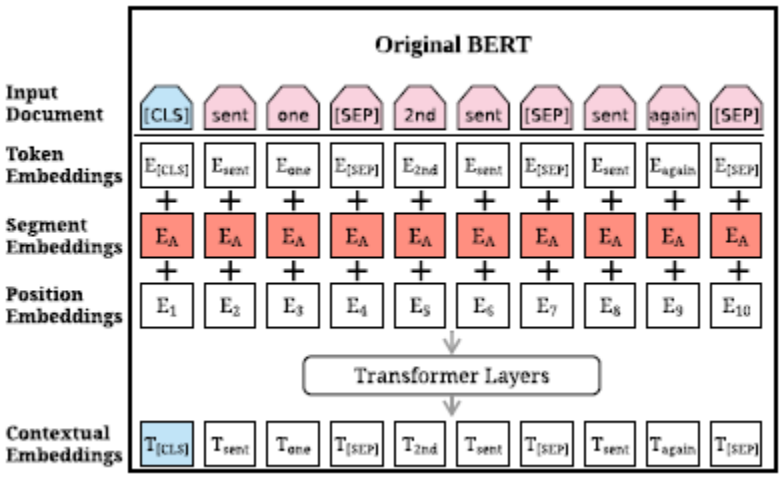


Figure 4.2 Original BERT

4.3.6.2 Modified BERT for Summarization Task

In order to represent individual sentences, we insert external [CLS] tokens at the start of each sentence, and each [CLS] symbol collects features for the sentence preceding it. We also use interval segment embeddings to distinguish multiple sentences within a document. For each sentence, we assign segment embedding EA or EB depending on whether it is odd or even. For example, for document [sent1; sent2; sent3; sent4; sent5], we would assign embeddings [EA;EB;EA;EB;EA]. This way, document representations are learned hierarchically where lower Transformer layers represent adjacent sentences, while higher layers, in combination with self-attention, represent multi-sentence discourse.

Moreover, Position embeddings in the original BERT model have a maximum length of 512; we overcome this limitation by adding more position embeddings that are initialized randomly and finetuned with other parameters in the encoder.

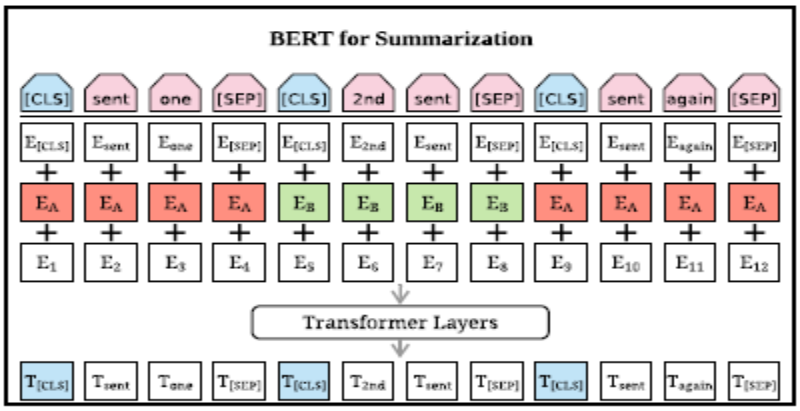


Figure 4.3 Modified BERT for Summarization

4.3.7 Full Model Architecture

The full model architecture consists of several layers that process the input text and generate the output. The model architecture can be described as follows:

1. **BERT Encoder**: The model starts with a BERT (Bidirectional Encoder Representations from Transformers) encoder, which is a pre-trained transformer-based model. The BERT encoder takes the input text and generates contextualized word representations. It uses multiple layers of self-attention mechanisms to capture the relationships between words in the input.

2. **Transformer Layers**: The BERT encoder is followed by additional transformer layers. These layers refine the contextualized word representations obtained from BERT and allow the model to capture more complex patterns and dependencies in the input text. Each transformer layer consists of multi-head self-attention and feed-forward neural network sub-layers.

3. **Sentence Scoring**: After the transformer layers, the model applies a sentence scoring mechanism. This step involves assigning a score to each sentence in the input document based on its importance for the summary through passing the contextualized word representations obtained from the transformer layers through a series of fully connected layers. The feed-forward network applies linear transformations and non-linear activation functions to capture complex relationships and patterns in the encoded representations.

4. **Sentence Selection**: Once the sentences are scored, the model selects the top-ranked sentences to form the summary. The number of selected sentences can be predetermined or based on a desired summary length.

In summary, the model utilizes a pre-trained BERT encoder, transformer layers, and a sentence scoring mechanism to generate extractive summaries. The model captures the contextual information of the input text and selects the most important sentences for the summary based on their scores. The specific architecture and configuration of the model can vary depending on the implementation and requirements of the summarization task.

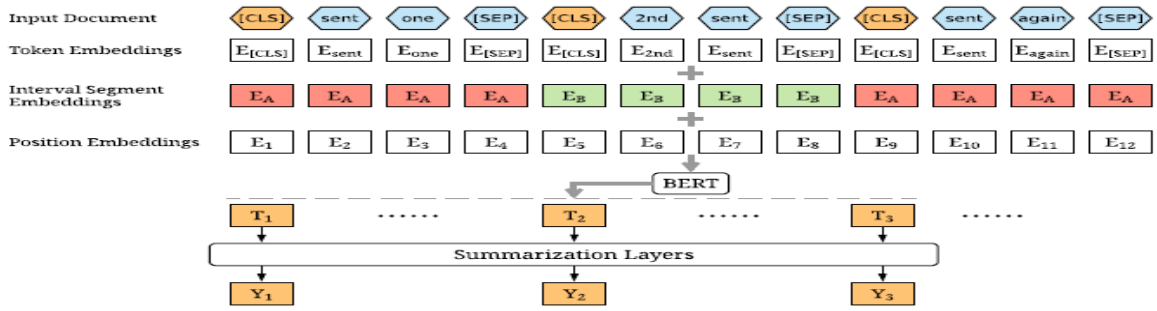


Figure 4.4 Full Model Architecture

4.3.8 Compression Ratio

The compression ratio module is a component of a text summarization system that allows the user to specify the desired compression ratio for generating summaries. The compression ratio refers to the reduction in the length or size of the original text achieved by the summarization process.

In this module, the user has the flexibility to choose the compression ratio based on their specific requirements and preferences. The compression ratio is typically defined as a numerical value that represents the desired ratio between the length of the summary and the length of the original text. For example, a compression ratio of 0.5 would indicate that the summary should be half the length of the original text.

When the user specifies the desired compression ratio, the system adjusts the summarization algorithm accordingly to generate summaries that meet the specified criteria. The algorithm aims to capture the most important information from the source text while ensuring that the summary length adheres to the desired compression ratio.

This example demonstrates the different compression ratio levels output:

Original Paragraph: "The team of scientists conducted a series of experiments to study the impact of climate change on marine ecosystems. They collected data from various coastal regions and analyzed the changes in sea temperature, salinity levels, and biodiversity. The findings revealed a significant decline in coral reefs and the migration patterns of marine species. The study also highlighted the increased risk of ocean acidification and its potential consequences for the ecosystem."

Summary (Compression Ratio 70%): "Scientists studied climate change's impact on marine ecosystems. Findings show declining coral reefs and species migration."

Summary (Compression Ratio 30%): "A team of scientists studied the impact of climate change on marine ecosystems by collecting and analyzing data from coastal regions. Their findings revealed a significant decline in coral reefs, changes in migration patterns of marine species, and increased risk of ocean acidification."

In the first summary with a compression ratio of 70%, only the most essential information is retained, resulting in a more concise summary. In the second summary with a compression ratio of 30%, more details are included, providing a broader overview of the study's findings. The choice of compression ratio allows the user to adjust the level of detail and conciseness in the generated summaries.

4.4 Abstractive Summarizer

Unlike extractive summarization, which selects and combines existing sentences, abstractive summarization involves a more creative process of generating new sentences that capture the essence of the source content. The abstractive summarization module employs advanced techniques such as neural networks, specifically encoder-decoder transformer models, to perform this task. The encoder-decoder architecture consists of two main components: the encoder and the decoder. The encoder processes the input text and learns a representation that captures the contextual information and semantic meaning of the text. The decoder, on the other hand, generates a summary by decoding the learned representation and producing coherent and concise sentences.

4.4.1 Functional Description

4.4.1.1 Input

In the context of extractive summarization, the input to the summarizer is a source document that contains multiple sentences or phrases. This source document can be a lengthy article, a news report, a research paper, or any other text that requires summarization. The input serves as the basis for extracting key information and identifying important sentences or phrases that capture the essence of the document.

4.4.1.2 Output

It is a generated set of new sentences that are not necessarily present in the original text. These sentences are creatively crafted by the model to convey the key information and main ideas of the source document in a condensed form. The output summary is intended to provide a comprehensive overview of the original content while maintaining readability and coherence.

4.4.1.3 Functionality

The functionality of the abstractive summarizer is as follows:

1. Input Processing: The abstractive summarizer takes a source document as input, which can be a lengthy text such as an article, blog post, or document. The input is typically a sequence of words or tokens.

2. Text Understanding: The summarizer employs natural language processing techniques to understand the content of the input text. It analyzes the syntactic structure, identifies important keywords, and extracts key information from the source document.

3. Information Compression: The abstractive summarizer compresses the information from the source document by generating a concise summary that captures the main points and key ideas. It aims to condense the content while maintaining the essential meaning.

4. Language Generation: The summarizer employs advanced language generation techniques, such as neural networks or transformer models, to generate human-like and coherent sentences. It uses learned patterns and language modeling to produce grammatically correct and contextually appropriate summaries.

5. Content Fusion: The abstractive summarizer combines information from different parts of the source document to create a coherent and cohesive summary. It may rephrase or paraphrase sentences, merge similar ideas, and synthesize concepts to ensure the summary captures the essence of the original content.

6. Output Presentation: The summarizer presents the generated summary as the output. The summary is typically a shorter version of the source document, highlighting the most important information. It aims to provide a comprehensive overview that enables readers to grasp the main points without reading the entire document.

Overall, the abstractive summarizer functions by understanding the input text, compressing, and synthesizing the information, generating coherent sentences, and presenting a concise summary that captures the essential content of the source document.

4.4.2 Modular Decomposition

1. Tokenization: The first step in the encoder is tokenization, where the input document is split into individual tokens or subwords. This is done using a tokenizer specific to BERT, which handles issues such as out-of-vocabulary words and word splitting. The tokenizer maps each token to its corresponding index in the vocabulary.

2. Input Embedding: Once the document is tokenized, the tokens are converted into dense vector representations called embeddings. These embeddings capture the semantic and contextual information of each token. In BERT, the input embeddings

are learned during the pre-training phase. The input embedding layer maps each token index to its corresponding embedding vector.

3. Positional Encoding: To preserve the positional information of the tokens in the document, BERT incorporates positional encoding. Positional encoding adds a fixed-length vector to each token's embedding, indicating its position in the sequence. This helps the model differentiate between tokens based on their relative positions and capture the sequential dependencies in the document.

4. Transformer Encoder Layers: The core component of the encoder is the stack of transformer encoder layers. Each encoder layer consists of two sub-layers: the multi-head self-attention mechanism and the position-wise feed-forward neural network.

    - Multi-Head Self-Attention: The self-attention mechanism allows the model to attend to different parts of the input document while capturing the relationships between tokens. It computes the importance of each token in the context of the entire document. BERT utilizes multi-head self-attention, which performs self-attention computation multiple times with different learned linear projections, allowing the model to focus on different aspects of the input.
    - Position-Wise Feed-Forward Network: After the self-attention sub-layer, a position-wise feed-forward neural network is applied to each token independently. This network consists of two linear transformations with a ReLU activation function in between. It helps to capture complex non-linear interactions between tokens and further refine the representations.

5. Layer Normalization: Layer normalization is applied after each sub-layer in the transformer encoder. It normalizes the values of the tokens within each layer, ensuring that the model can learn stable and effective representations. It helps in reducing the internal covariate shift and enables faster training convergence.

4.4.3 Design Constraints

The design constraints of abstractive summarization include:

1. Language Fluency: The generated summary should exhibit fluency and readability, with grammatically correct sentences and appropriate use of vocabulary. The summarizer needs to consider the target audience and generate summaries that are understandable and coherent [6].

2. Content Preservation: The summarizer should strive to preserve the important content and key ideas from the source document. It should avoid omitting critical information or distorting the original meaning while condensing the text.

3. Length Constraint: Abstractive summarization often involves specifying a maximum length or word limit for the summary. This constraint ensures that the

generated summary remains concise and focused. It requires the summarizer to prioritize the most relevant information and avoid excessive verbosity [6].

4. Contextual Understanding: The summarizer should have the ability to understand the contextual nuances and meaning of the source document. It should be able to capture the relationships between sentences, infer implied information, and generate summaries that reflect the overall context.

5. Generalization Capability: The summarizer should have the capability to generalize and handle various types of source documents across different domains or topics. It should be able to adapt and generate accurate summaries for diverse content, without being overly specific to a particular type of input.

These design constraints play a crucial role in developing an effective and reliable abstractive summarization system that produces high-quality summaries while considering factors such as language fluency, content preservation, length constraint, coherence, context understanding, and generalization capability.

4.4.4 Input dataset

The CNN/Daily Mail dataset is a widely used dataset in the field of abstractive summarization. It is a large-scale dataset that contains news articles paired with human-written summaries. The dataset is collected from the websites of CNN and the Daily Mail, covering a diverse range of topics and news articles.

4.4.4.1 Preparing the dataset for the Abstractive Summarization Task

The process of preparing the CNN/Daily Mail dataset for the abstractive summarization task involves several steps.

1. **Data Cleaning**: Raw text data often contains noise, inconsistencies, and irrelevant information. Data cleaning involves removing any unnecessary characters, symbols, or formatting issues that may interfere with the training process. It may also involve removing duplicates, correcting errors, and standardizing the text format.

2. Special Tokens: Adding special tokens at the start and end of the summarized text is a common practice in abstractive summarization models. These special tokens serve as markers to indicate the beginning and end of the summary, allowing the decoder to identify the boundaries of the generated output. Here are examples of how the start and end tokens are added to the summarized text:

    - Original summarized text: "The researchers discovered a new species of butterfly."
    - Summarized text with start and end tokens: "[CLS] The researchers discovered a new species of butterfly. [SEP]"

The logic behind this approach is to leverage the ROUGE scores as a measure of the relevance and importance of sentences in the document summary. By selecting sentences with higher ROUGE scores, we prioritize those that have a stronger overlap with the reference summary, indicating their importance in capturing the essence of the original document. The greedy selection algorithm iteratively adds sentences to the summary based on their ROUGE scores, gradually constructing a concise summary that represents the key information from the source document.

Overall, this process combines the calculation of ROUGE scores, the application of a greedy selection algorithm, and the cleaning of sentences to extract the most relevant and informative sentences from the CNN/Daily Mail dataset for the extractive summarization task.

4.4.5 Tokenization and word embedding

The abstractive summarizer uses a predefined vocabulary, which is a set of all possible words in the input and output space. Tokenization is the process of splitting the input text and summary text into individual tokens or subwords. This enables the model to process the text at a granular level.

4.4.5.1 Tokenization

Tokenization involves breaking down the input text into a sequence of subword tokens. Unlike traditional tokenization, which splits text at the word level, our model tokenizes text into subword units using a technique called WordPiece tokenization. This process allows us to handle out-of-vocabulary words and capture more fine-grained information.

During tokenization, special tokens such as [CLS] and [SEP] are added to mark the beginning and separation of sentences or documents. The input text is then split into fixed-length segments or truncated to fit the model's input length limit.

4.4.5.2 Word Embedding

Word Embedding: Word embedding in the extractive summarization module is achieved using pre-trained BERT (Bidirectional Encoder Representations from Transformers) models. BERT models are trained on large amounts of text data and learn contextualized representations of words.

4.4.6 Full Model Architecture

The full model architecture consists of several layers that process the input text and generate the output. The model architecture can be described as follows:

1. **Encoder**: The encoder component processes the input document and encodes it into a contextualized representation. This can be achieved using encoder-only transformer which captures the contextual information of the input text.

2. **Decoder**: The decoder component generates the summary based on the encoded representation produced by the encoder. It utilizes an autoregressive language model, such as a recurrent neural network (RNN) or a transformer, to sequentially generate the summary tokens. The decoder attends to the encoded representation to extract relevant information while generating each token.

3. **Attention Mechanism**: The attention mechanism is a crucial part of the abstractive summarizer. It allows the model to focus on different parts of the input document when generating the summary. It assigns weights to different words or phrases in the input based on their relevance to the current summary token being generated.

4. **Embedding Layer**: The embedding layer converts the input tokens and summary tokens into dense vector representations known as word embeddings. These embeddings capture the semantic and contextual information of the words and help the model understand the relationships between them.

4.4.7 Beam Search Decoding

During the inference or generation phase, beam search is commonly employed to explore multiple possible summary sequences and select the most coherent and relevant one. It maintains a beam of the top-K candidate sequences and expands them based on the model's predictions, ultimately selecting the sequence with the highest probability. Beam search avoids being totally greedy while keeping the search space smaller than exhaustive search. A typical value of k ranges between 5 to 10.

The goal of the decoder is to maximize the probability of the output sequence for the given input sequence. The problem with greedy decoding is that choosing the word with the highest probability at each time step does not guarantee the maximum probability over the whole sequence. In order to find the optimum solution, we should generate all the possible sequence combinations and choose the sequence with the highest probability, but this is very expensive as the search space is very large. To reach a better solution for the decoding problem beam search technique was introduced.

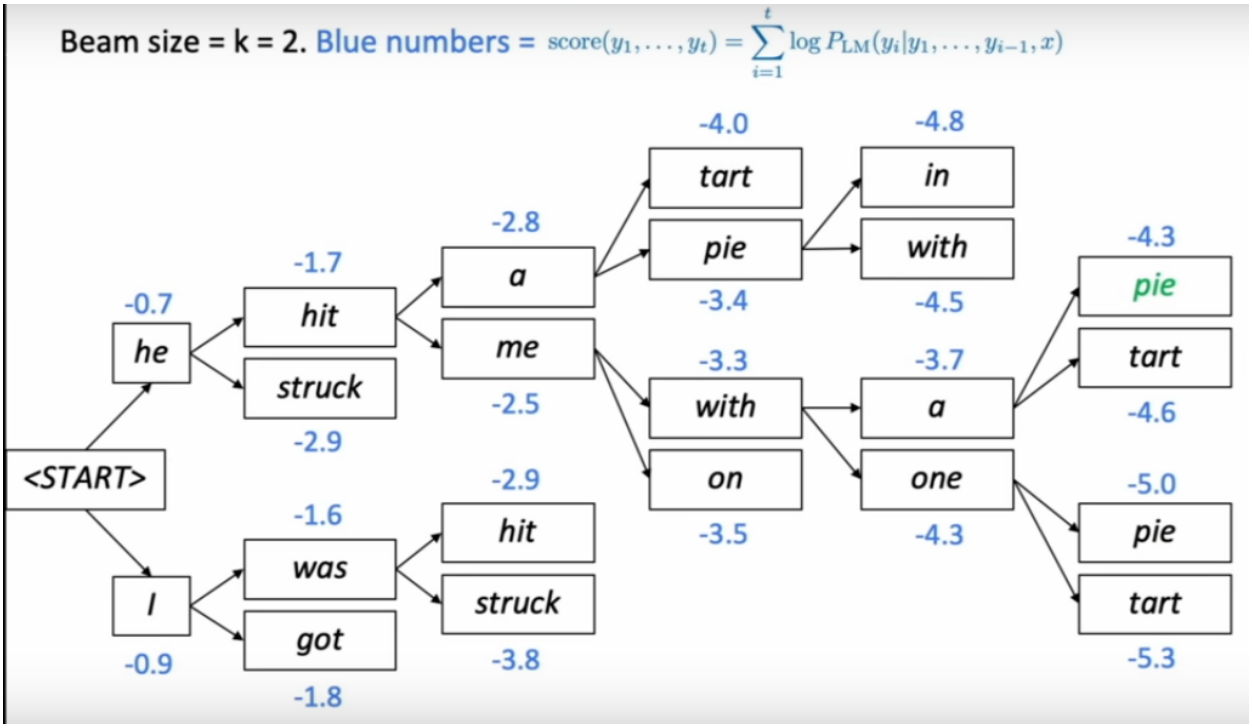


Figure 4.5 Example of Beam Search with k = 2

4.4.8 Coverage and Length Penalties

The length penalty function encourages the generation of summaries that are neither too short nor too long. It is designed to address the bias towards shorter sequences that often occurs in beam search. A common approach is to apply a penalty that increases as the length of the generated sequence increases. This discourages the model from producing overly lengthy summaries and promotes the generation of concise and informative outputs.

The coverage penalty function aims to encourage the summarization model to cover important information from the source text. It penalizes the repetition of words or phrases in the generated summary. By discouraging redundant information, the coverage penalty helps ensure that the summary provides a comprehensive overview of the key points in the source text. This is particularly important in abstractive summarization, where the model has the flexibility to generate new sentences and should avoid excessive repetition.

4.5 Text-to-Emotion

The text-to-emotion module that we have developed follows a specific architecture to analyze and classify emotions expressed in text data. The architecture comprises several key components that work together to process the input text and generate emotion predictions. Here is an overview of the module's architecture:

The architecture of our text-to-emotion module comprises several key components. It begins with an embedding layer that converts textual input into vector representations, allowing the neural network to understand the underlying meaning of the words. Following the embedding layer, we have three Long Short-Term Memory (LSTM) layers. LSTMs are recurrent neural networks that excel in capturing sequential dependencies and context in text data. These layers enable the model to capture the intricate patterns and contextual information necessary for accurately inferring emotions from text. After the LSTM layers, we have a dense layer that applies a linear transformation to the LSTM output, allowing for further feature extraction and integration. This dense layer aids in mapping the learned representations to the desired output of emotion classification. Together, these components form a comprehensive architecture that effectively translates text input into predicted emotions, enabling our text-to-emotion module to understand and analyze the emotional content conveyed within textual data [17].

4.5.1 Functional Description

The text-to-emotion module analyzes and classifies emotions in text data. It processes the input text by converting it into word embeddings and then passes it through three LSTM layers to capture sequential dependencies and context. Features related to emotions are extracted, followed by a dense layer that applies a linear transformation. The module generates predictions for the expressed emotions based on the transformed features. In summary, the module takes text as input, processes it through LSTM layers to extract emotional features, and produces emotion predictions using a dense layer.

4.5.2 Modular Decomposition

Embedding Layer:

The first component of our architecture is the embedding layer. It takes the input text and converts it into a numerical representation known as word vectors or embeddings. These embeddings capture semantic and contextual information of words in the text. This layer helps in transforming the text data into a format that can be processed by subsequent layers of the model.

LSTM Layers:

Following the embedding layer, our architecture includes three Long Short-Term Memory (LSTM) layers. LSTMs are recurrent neural networks (RNNs) that are well-suited for capturing sequential dependencies in text data. Each LSTM layer processes the

sequential input from the previous layer and extracts relevant features. The hidden states within the LSTM layers enable the model to retain important information over longer sequences, aiding in understanding the context and sentiment expressed in the text.

Dense Layer:

After the LSTM layers, our architecture incorporates a dense layer. The dense layer is a fully connected layer that applies a linear transformation to the output from the LSTM layers. It performs a weighted sum of the input features and applies an activation function to produce the final emotion predictions. The dense layer helps in mapping the extracted features to the specific emotion categories.

4.5.3 Design Constraints

When designing our text-to-emotion model, we encountered several design constraints that influenced our choices. Firstly, we had to consider the computational resources available for our model. Since we aimed for broad accessibility, we optimized the model to run efficiently on various devices, including resource-constrained platforms like mobile phones. Secondly, the availability and quality of training data posed a constraint. To mitigate this, we implemented data augmentation techniques and explored transfer learning to address limited or biased training data. Additionally, we had to ensure that our model met specific response time requirements for real-time applications. We optimized our algorithms and performed model inference optimizations to achieve low-latency processing. Interpretability was another consideration, as some use cases required explanations for predicted emotions. To address this, we incorporated attention mechanisms and other explainable AI techniques.

4.5.4 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.

4.5.5 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.

4.5.6 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.

4.5.7 Tokenization and word embedding

4.5.7.1 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.

4.5.7.2 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.

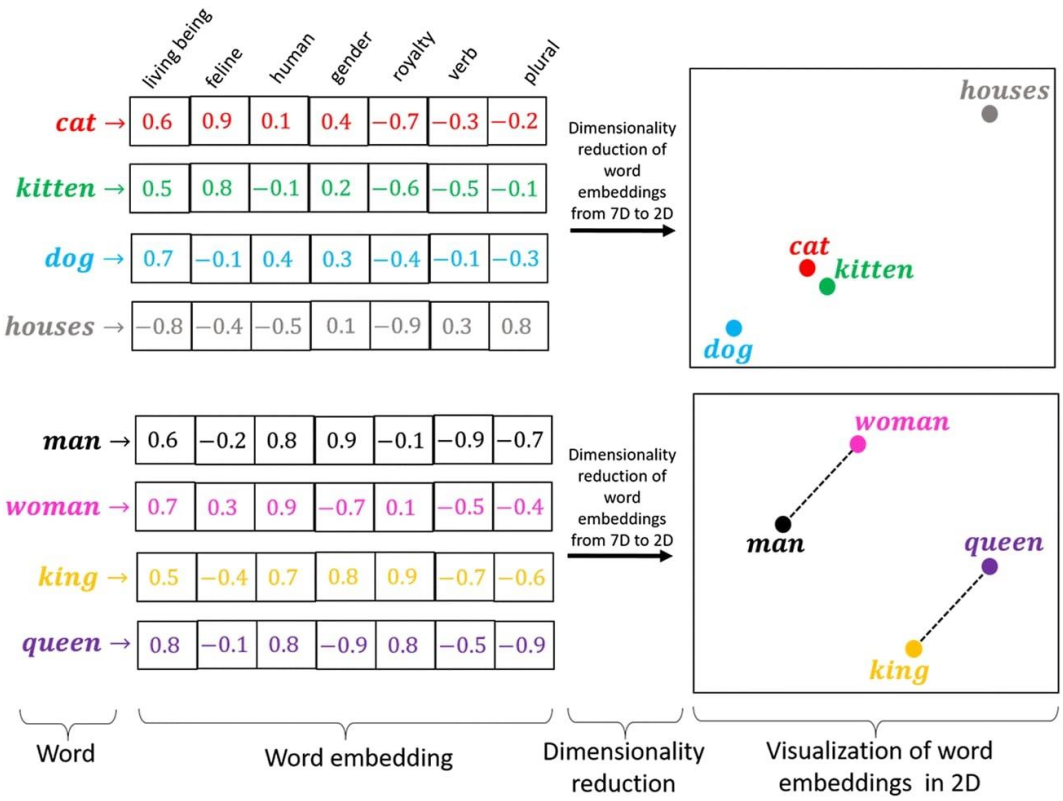


Figure 4.6 GloVe Word Embedding

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.

4.5.8 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 represent 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 ;it applies the SoftMax activation function, 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.

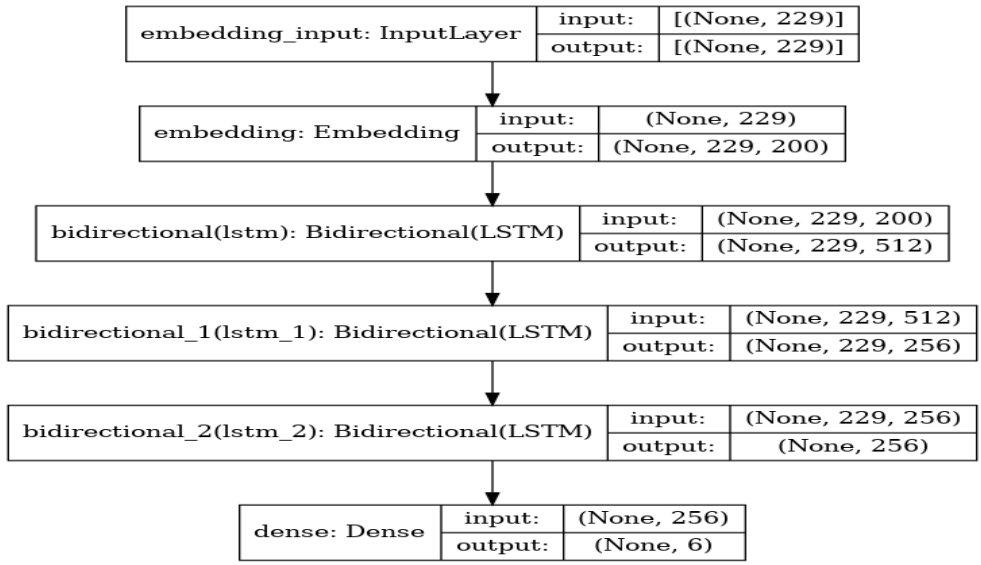


Figure 4.7 Text-to-Emotion Model Architecture

4.5.9

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

Table 4.1 Sentiment Analysis result

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 [18].

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.

4.6 News Genre Classifier

4.6.1 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.

4.6.2 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.

4.6.3 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.

4.7 Text to Speech

4.7.1 Functional Description

Tacotron is a text-to-speech model used in the Speak News application. It converts written text into natural-sounding speech by utilizing deep learning techniques. The model takes textual input as well as linguistic and acoustic features and generates corresponding spectrograms. These spectrograms are then converted into audio waveforms, resulting in high-quality synthesized speech. Tacotron utilizes recurrent neural networks (RNNs) and attention mechanisms to capture the context and prosody of the input text, enabling it to produce expressive and intelligible speech output. The model's flexibility and ability to handle various languages and accents make it a powerful tool for creating realistic synthetic voices in the Speak News app.

4.7.2 Modular Decomposition

Text Encoding Module:

This module takes the input text and performs preprocessing tasks such as text normalization and tokenization. Then it converts the tokenized text into a sequence of phonetic or linguistic features, capturing the linguistic content of the input text. These features are passed to the next module for further processing.

Acoustic Encoding Module:

The acoustic encoding module takes the linguistic features from the previous module and extracts additional acoustic features. These features can include prosodic information, such as pitch and duration, which play a crucial role in generating natural-sounding speech. The acoustic features are combined with linguistic features to provide a comprehensive representation of the input text.

Decoder Module:

The decoder module utilizes a recurrent neural network (RNN) to decode the combined linguistic and acoustic features into a sequence of spectrograms. It considers the contextual information from the previous steps and generates spectrograms that capture the temporal variations in speech. The decoder module uses an attention mechanism to focus on relevant parts of the input during the spectrogram generation process.

Spectrogram-to-Waveform Synthesis Module:

After obtaining the spectrogram sequence, the spectrogram-to-waveform synthesis module converts it into a time-domain waveform, which represents the synthesized speech. Griffin-Lim algorithm is commonly used for this purpose, which iteratively reconstructs the waveform from the spectrogram. This module ensures that the generated waveform aligns with the desired acoustic properties, such as the pitch contour and phonetic duration.

4.7.3 Design Constraints

Computational Resources:

Tacotron requires significant computational resources, especially during the training phase, due to the complex nature of deep learning models. The size of the model and the amount of data used for training can pose constraints on the available computational power, memory, and processing time. Designing Tacotron to efficiently utilize available resources is crucial to ensure its practicality and scalability.

Data Availability and Quality:

The performance of Tacotron heavily relies on the availability and quality of the training data. Sufficient and diverse training data that covers a wide range of languages, accents, and speech styles is essential for the model to generalize well. Constraints arise when data for specific languages or domains is scarce or of low quality, which can affect the model's ability to produce accurate and natural-sounding speech.

Model Size and Latency:

Tacotron's model size affects both storage requirements and inference latency. Large models may consume substantial storage space, limiting the deployment options, particularly in resource-constrained environments such as mobile devices. Additionally, the latency or response time of Tacotron during real-time inference needs to be minimized to provide a seamless user experience.

Robustness to Variations and Ambiguities:

Tacotron should be designed to handle variations in input text, such as different sentence structures, punctuation, and abbreviations. The model should be robust enough to handle ambiguous text inputs and provide coherent and contextually appropriate speech output. Designing Tacotron to handle such variations and ambiguities requires careful consideration of the architecture, training techniques, and data augmentation strategies.

Naturalness and Intelligibility:

Tacotron should generate speech that is both natural-sounding and intelligible to the listeners. Balancing these two aspects can be challenging, as overly natural-sounding speech may sacrifice intelligibility, particularly for non-native speakers or in noisy environments. Designing Tacotron to strike the right balance between naturalness and intelligibility is crucial for its usability and effectiveness in real-world applications.

4.7.4 Input

The input of the Tacotron model consists of textual data, typically in the form of written sentences or paragraphs. The text is preprocessed and encoded to capture linguistic and acoustic features, providing context and prosody information. These features are used by the model to generate spectrograms, which serve as the intermediate representation for synthesizing speech.

4.7.5 Input Dataset

The input dataset for Tacotron, specifically using the LJ Speech dataset, consists of a collection of audio recordings and corresponding text transcripts. The dataset comprises a diverse set of English speech samples, including a single speaker reading sentences from various sources. Each audio recording is paired with a corresponding text transcript, allowing for supervised training of the Tacotron model. The text transcripts provide the linguistic content for the model, while the audio recordings serve as the ground truth for training. By using the LJ Speech dataset, Tacotron can learn the relationship between the textual input and the desired speech output, enabling it to generate synthesized speech accurately.

4.7.6 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.

2. 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).

3. 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.

4. 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.

5. 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.

4.7.7 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.

    a. 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.

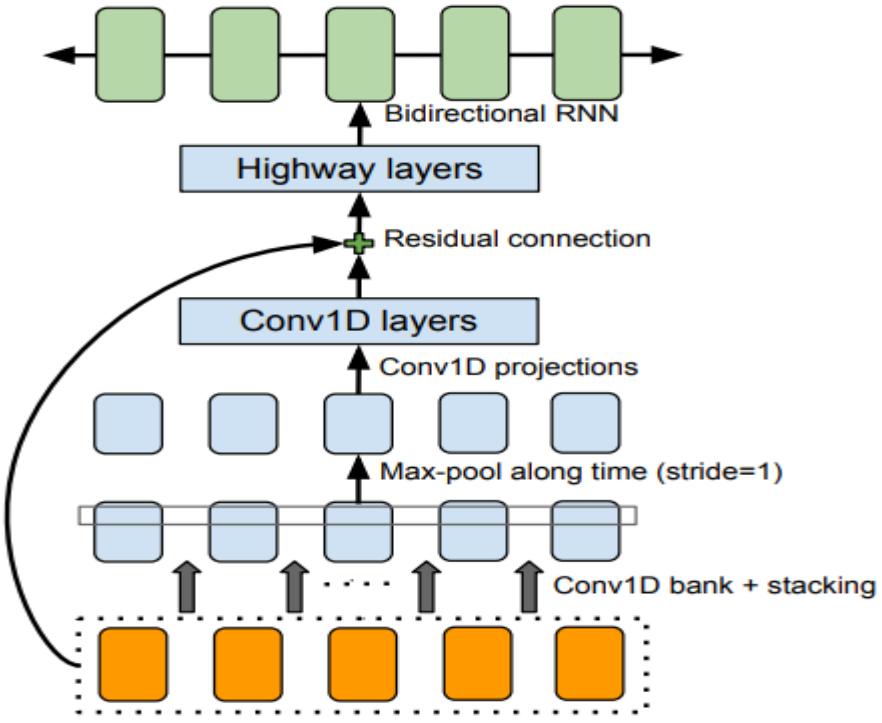


Figure 4.8 Encoder of the Text to speech

2. 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.

3. 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.

4. Post-processing: The generated mel-spectrograms may undergo post-processing steps, such as smoothing or dynamic range compression, to improve their quality and naturalness.

5. 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.

Spectral analysis	*pre-emphasis*: 0.97; *frame length*: 50 ms; *frame shift*: 12.5 ms; *window type*: Hann
Character embedding	256-D
Encoder CBHG	*Conv1D bank*: K=16, conv-k-128-ReLU *Max pooling*: stride=1, width=2 *Conv1D projections*: conv-3-128-ReLU $\rightarrow$ conv-3-128-Linear *Highway net*: 4 layers of FC-128-ReLU *Bidirectional GRU*: 128 cells
Encoder pre-net	FC-256-ReLU $\rightarrow$ Dropout(0.5) $\rightarrow$ FC-128-ReLU $\rightarrow$ Dropout(0.5)
Decoder pre-net	FC-256-ReLU $\rightarrow$ Dropout(0.5)$\rightarrow$ FC-128-ReLU $\rightarrow$ Dropout(0.5)
Decoder RNN	2-layer residual GRU (256 cells)
Attention RNN	1-layer GRU (256 cells)
Post-processing net CBHG	*Conv1D bank*: K=8, conv-k-128-ReLU *Max pooling*: stride=1, width=2 *Conv1D projections*: conv-3-256-ReLU $\rightarrow$ conv-3-80-Linear *Highway net*: 4 layers of FC-128-ReLU *Bidirectional GRU*: 128 cells
Reduction factor (r)	2

Figure 4.9 Text to speech full architecture

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.

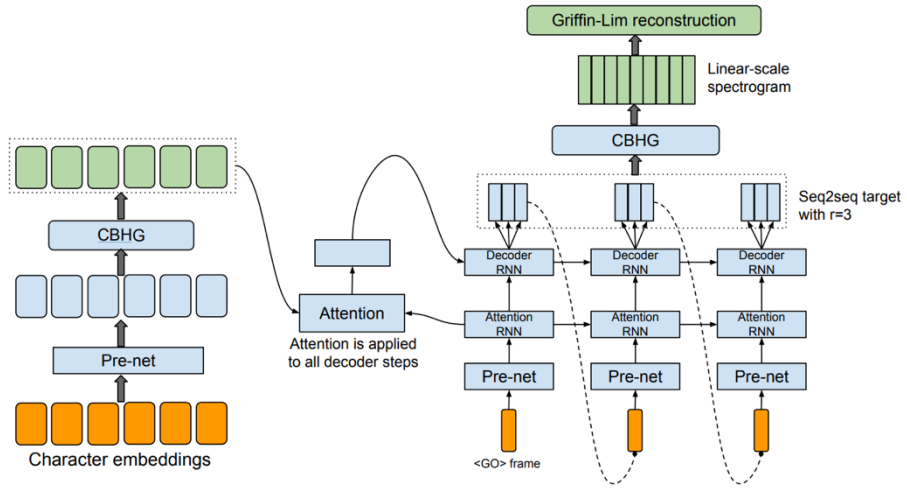


Figure 1: *Model architecture. The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech.*

Figure 4.10 Text to speech full architecture diagram

4.8 Avatar Generation

This paragraph is an overview of this module. The avatar generation module takes the photo or video as the face of the avatar to be generated, and it takes audio input which is the output of the text to speech. These two inputs pass through the face decoder to reconstruct the face sequence with the lip synced. After that, the output of the decoder passes through a lip sync discriminator to detect the real from fake lip sync. In summary, it takes input audio and video/photo and produce a video of the avatar.

4.8.1 Functional Description

The first step in the module is to take the audio signal and extract the features like MFCCs, and spectral characteristics. During the first step the face encoder will extract the lip features from a given photo or video such as the lip contours and landmarks using computer vision techniques. After the face encoder, the extracted lip features are aligned with the corresponding phenome-viseme, and phonetic information obtained from the speech analysis stage. Why are we doing this? To obtain temporal correspondence between speech and lip movements.

We have lip sync model which uses machine learning to learn the mapping between the speech features and lip features. We trained the model on LRS2 dataset [20] from the BBC where it learns to generate appropriate lip movements given the input speech features. The lip generation, lip movements can be visualized by animating a 3D avatar or by directly manipulating the lip features of a target image or video.

Post-Processing the generated lip movements may undergo post-processing to enhance the quality, smoothness, or realism of the output. This involves techniques filtering, smoothing, or adding subtle facial expressions to improve the naturalness of the lip movements.

4.8.2 Modular Decomposition

The first step in our cycle is to preprocess the dataset as it consists of videos. A sequence of frames and audio files were extracted from the video to be used in the training. During the extraction of the frames the relevant lip features were extracted.

The architecture of the module consists of a face encoder, audio encoder, and face decoder. The face encoder blocks are used to process the face sequence. Each block is a sequence of convolutional layers that down sample the input. The blocks progressively reduce the spatial dimensions of the input while increasing the number of channels. The audio encoder processes the audio sequence. It consists of several convolutional layers that extract features from the audio waveform. Like the face encoder blocks, the audio encoder reduces the spatial dimensions and increases the number of channels. The face decoder blocks take the encoded audio features and decode them to generate a reconstructed face sequence. The decoder blocks are similar to the encoder blocks but perform up sampling instead of down sampling. They progressively increase the spatial dimensions while decreasing the number of channels.

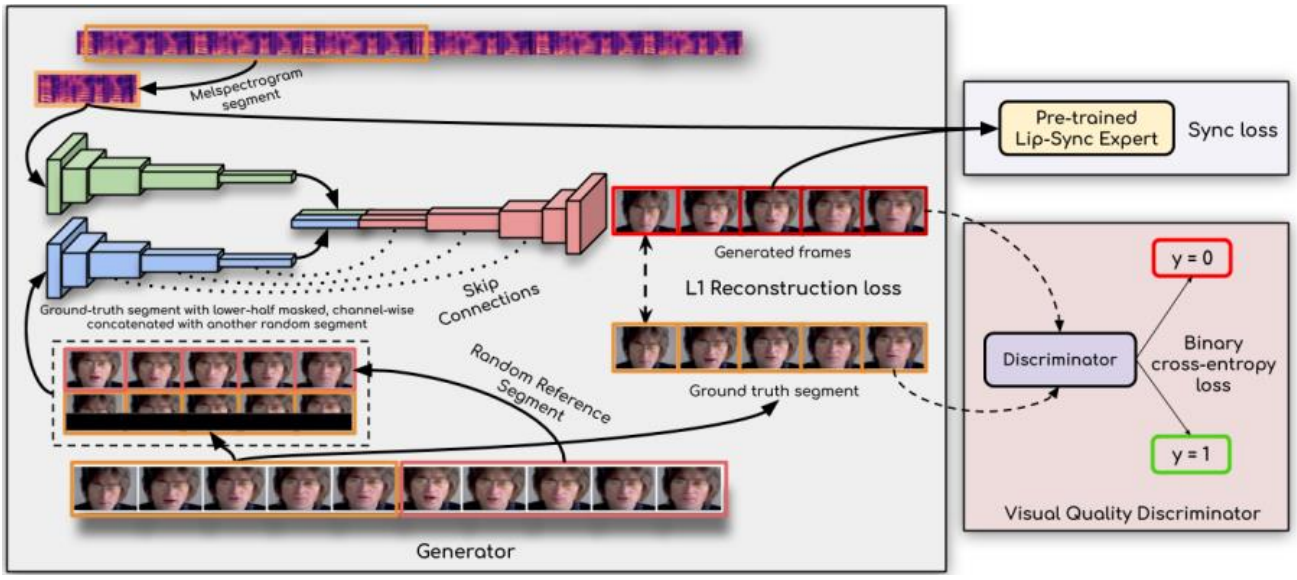


Figure 4.11 The architecture of the wav2lip diagram [20]

The output block is the final part of the module. It takes the output of the face decoder blocks and applies a series of convolutional layers to generate the final output. The output is a sequence of face images with three channels (RGB). The Sigmoid activation function is applied to the output to ensure the pixel values are between 0 and 1. The forward method of the module performs the actual forward pass computation. It takes the audio sequences and face sequences as input. It first processes the audio sequences through the audio encoder to obtain audio embeddings. Then, it iterates over the face encoder blocks and decoder blocks, passing the face sequences and audio embeddings through them. The intermediate feature maps from the encoder blocks are stored in a list. The output of each decoder block is concatenated with the corresponding feature map from the encoder blocks. Finally, the output block is applied to generate the final output sequence. The code handles input sequences of different dimensions by checking the input dimension size. If the input sequences have more than four dimensions, it assumes the input is a batch of sequences and performs additional reshaping and splitting operations. Otherwise, it returns the output as is.

4.8.3 Design Constraints

1. **Real-time performance**: we should focus on optimizing the inference of generating the avatar to enable real time performance. This requires efficient processing of audio and lip extracted from videos. Methods that could be implemented:

    a) Parallelization
    b) Hardware acceleration

2. **Robustness to Audio Variations**: we should design the model to handle a wide range of audio variations commonly encountered in real-world scenarios. Including variations in accents, speech rates, and background noise. Robust audio preprocessing techniques and acoustic modeling should be employed to ensure accurate synchronization between audio and lip movements, even in challenging audio conditions.

3. **Lip Movement Accuracy**: The model should be capable of capturing the intricate details and dynamics of lip movements accurately. Architectural choices, such as the use of RNNs or attention mechanisms, can help improve the model's ability to generate realistic and precise lip movements.

4. **Generalization to Diverse Speakers**: The model should be trained and designed to generalize well across a diverse range of speakers. It should account for variations in facial structures, shapes, and appearances. Techniques such as data augmentation, domain adaptation, or speaker adaptation can be employed to ensure the model's ability to generate lip movements that are suitable for a wide range of individuals.

5. **Scalability and Efficiency**: The design should consider scalability during the training process. This involves optimizing the training pipeline to handle large-scale

datasets efficiently, enabling the model to leverage a vast amount of data for improved performance. For inference, the model should be optimized to minimize computational resource requirements while maintaining high performance, allowing it to be deployed on various devices with limited resources.

6. **Ethical Considerations**: The design should address ethical considerations associated with the usage of the model. This includes prioritizing user privacy by incorporating data anonymization techniques and ensuring compliance with relevant data protection regulations. Additionally, the model should be developed and deployed in a manner that respects user consent and provides clear information on how the audio and video data will be used, stored, and shared.

Chapter 5 System Testing and Verification

5.1 Testing Setup

The testing setup plays a crucial role in ensuring the performance, accuracy, and reliability of a comprehensive news consumption solution, consisting of a news summarizer, text-to-speech model, and generated avatar for news narration. Through a well-designed testing process, the system's functionality, usability, and overall user experience are thoroughly evaluated to deliver a high-quality and seamless news consumption experience.

The testing process begins with the collection and preparation of a diverse dataset of news articles, encompassing a wide range of topics, writing styles, and lengths. This dataset serves as the foundation for training the models and validating their performance. By splitting the dataset into training, validation, and testing subsets, the effectiveness of the system can be objectively assessed.

During training, each component of the system is trained separately using the training dataset. The news summarizer model learns to extract essential information and generate concise summaries, while the text-to-speech model is trained to convert textual information into natural-sounding speech. Simultaneously, the generated avatar model is trained to animate an avatar with appropriate lip sync and facial expressions based on the provided text. The validation dataset is employed to fine-tune and optimize the models' performance, ensuring they meet the desired standards. Data Collection and Preparation: Gather a diverse dataset of news articles from various reliable sources. Ensure that the dataset covers different topics, writing styles, and lengths. Preprocess the data by removing unnecessary characters, formatting inconsistencies, and outliers to ensure data quality and consistency.

The Testing setup had many steps which were:

1. **Training Data Split**: Divide the dataset into training, validation, and testing subsets. The training set is used to train the models, the validation set helps in fine-tuning hyperparameters and monitoring performance during training, and the testing set is reserved for final evaluation.

2. **Text Summarizer Training**: Train the news summarizer model using the training dataset. The model should learn to extract important information and generate concise summaries. Evaluate the model's performance on the validation set, adjusting as needed.

3. **Text-to-Speech Model Training**: Train the text-to-speech model using the training dataset. The model should learn to convert text into natural-sounding speech. Monitor the model's performance on the validation set, refining the training process if necessary.

4. **Generated Avatar Training**: Train the generated avatar model using the training dataset. The model should learn to animate an avatar with appropriate lip sync

based on the text input. Validate and fine-tune the model's performance on the validation set.

5. **Text-to-Emotion Module**: Develop and train a text-to-emotion model using techniques such as sentiment analysis or emotion recognition. Evaluate the module's accuracy and effectiveness in detecting the emotional content of the news articles by measuring metrics such as accuracy, precision, recall, and F1-score.

6. **News Genre Classification**: Train a news genre classification model using techniques like supervised learning or deep learning. Evaluate the model's performance on the test set by measuring metrics such as accuracy, precision, recall, and F1-score.

7. **Integration Testing**: Combine all the modules to create an end-to-end system. Test the integration of the news summarizer, text-to-speech model, generated avatar, text-to-emotion, and news genre classification modules, ensuring seamless communication and compatibility between each component.

5.2 Testing Plan and Strategy

Our test plan and strategy involve a comprehensive approach to validate the functionality, performance, and usability of the news summarizer, text-to-speech model, generated avatar, text-to-emotion, and news genre classification modules. We will conduct unit testing to verify the individual functionality of each module, integration testing to ensure seamless communication between components, and functional testing to validate the system's end-to-end workflow.

5.2.1 Module Testing

5.2.1.1 Text summarization testing

In the field of text summarization, testing the effectiveness and quality of summarization systems is crucial. One widely used evaluation metric is ROUGE (Recall-Oriented Understudy for Gisting Evaluation). ROUGE is a set of metrics that measure the overlap between system-generated summaries and reference summaries.

ROUGE includes measures to automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans. It ranges from 0 to 1 where 0 is the lowest score and 1 the highest. There are four major variations of ROUGE Score: ROUGE-N, ROUGE-L, ROUGE-W, ROUGE-S. In our work, we used both ROUGE-N and ROUGE-L for assessment.

ROUGE-N measures the n-gram recall between the system summary and the reference summary. This is done by dividing the clipped count of the matching n-grams between the system and the reference summary by the total count of n-grams in the reference summary. The clipped count means that if the n-gram occurs in both the system and the

reference, only the minimum count of this n-gram in either the reference or the system is included in the following equation.

$$Rouge - N = \frac{\sum_{m}^{matching\ ngrams} Count_{clipped}(ngrams)}{\sum_{n}^{reference\ ngrams} Count(ngrams)}$$

ROUGE-L accounts for the similarity between the reference and the system's summary using the longest common subsequence between them.

$$Rouge - L = \frac{(1 + \beta^2) R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}}$$

Where LCS is the longest common subsequence

$$R_{lcs} = \frac{LCS(reference\ summary, system\ summary)}{length\ of\ reference\ summary}$$

$$P_{lcs} = \frac{LCS(reference\ summary, system\ summary)}{length\ of\ system\ summary}$$

Model	R1↑	R2↑	RL↑
Extractive Summarizers			
BERTSUMEXT	43.25	20.24	39.63
BERTSUMEXT w/o interval embeddings	43.20	20.22	39.59
Abstractive Summarizers			
TransformerABS	40.21	17.76	37.09

Table 5.1 Text Summarizer evaluation

5.2.1.2 Text to Speech testing

In the testing phase of the Speak News App, a comprehensive evaluation was conducted to assess the quality and performance of the text-to-speech (TTS) functionality. One of the evaluation methods employed was the Mean Opinion Score (MOS), a widely used subjective measure to gauge the perceived quality of synthesized speech.

Mean Opinion Score (MOS) is a subjective rating scale that allows human evaluators to provide their opinion on the quality of synthesized speech. MOS testing involves presenting a set of audio samples to a group of evaluators who rate each sample based on a predefined scale, typically ranging from 1 to 5. The scale is usually associated with labels such as "Bad," "Poor," "Fair," "Good," and "Excellent," reflecting different levels of perceived speech quality.

To conduct the MOS testing for the Speak News App, a diverse set of news summaries was selected as representative samples of the synthesized speech. These samples were chosen to cover various linguistic patterns, sentence structures, and content genres encountered in news articles. The selected news summaries were then converted into speech using the app's text-to-speech functionality.

We found out the following results comparing Tacotron to other models.

Model	Mean Opinion Score
Tacotron	3.82 ± 0.085

Table 5.2 Text to speech evaluation

By utilizing the Mean Opinion Score (MOS) testing methodology, the Speak News App's text-to-speech functionality was thoroughly evaluated, providing a measure of the perceived quality of the synthesized speech. The MOS scores obtained from human evaluators' ratings contributed to the assessment of the TTS system's performance and guided enhancements to ensure a satisfactory user experience in terms of speech quality and naturalness.

5.2.1.3 Text to Emotion

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

5.2.1.4 Avatar Generation

The testing and evaluation of the generated avatar for news narration were conducted to assess its quality and performance. Various metrics were employed to measure the effectiveness of the avatar generation process. These metrics included evaluating the visual representation and synchronization of the avatar's lip movements with the corresponding speech, assessing the believability and naturalness of the avatar's facial expressions, and gauging the overall user perception and engagement with the avatar's narration. Through rigorous evaluation, we aimed to ensure that the generated avatar effectively conveyed the news content and provided an immersive and engaging experience for users.

During the evaluation of the generated avatar for news narration, several metrics were employed to assess the quality and performance of the model. The first metric, Lip-Sync Error-Confidence, measures the accuracy of lip movements synchronization between the avatar and the speech. A lower Lip-Sync Error-Confidence score indicates a higher level of accuracy in matching the avatar's lip movements with the spoken words.

The second metric, Lip-Sync Error-Distance, quantifies the spatial discrepancy between the avatar's lip movements and the actual speech. It measures the Euclidean distance between the positions of the avatar's lip landmarks and the corresponding locations in the speech data. A lower Lip-Sync Error-Distance signifies better alignment between the avatar's lip movements and the speech.

The third metric used in the evaluation is Fréchet Inception Distance (FID), which assesses the visual quality and realism of the generated avatar. FID measures the similarity between the distribution of features extracted from real images and the generated avatar images. A lower FID score indicates that the generated avatar images are visually closer to real images and exhibit a higher degree of realism.

By employing these metrics during the evaluation process, we aimed to ensure that the generated avatar achieved accurate lip synchronization, minimal spatial discrepancy, and high visual quality, ultimately providing a compelling and immersive news narration experience for users.

These were our results for the avatar generation module:

LSE-D↓	**LSE-C ↑**	**FID↓**
9.53	6.41	13.65

Table 5.3 Avatar generation evaluation [20]

5.2.2 Integration test

During the integration testing of our full module, our focus is on achieving seamless communication and compatibility between the news summarizer, text-to-speech model, generated avatar, text-to-emotion, and news genre classification components. The integration tests encompass verifying input-output consistency, ensuring data integrity and compatibility, validating functional flow in real-world scenarios, testing error handling and recovery mechanisms, evaluating performance and scalability, and assessing the overall user experience. Through these tests, we aim to ensure that the integrated module operates smoothly, delivering accurate news summarization, natural speech synthesis, immersive avatar narration, reliable emotion analysis, and precise genre classification, resulting in a seamless and engaging news consumption experience for users.

5.3 Test Schedule

February 2023, we initiated our test plan for the machine learning models as part of our project

April 2023, the base models were successfully implemented and prepared for the planned testing phase. Over the course of one month, we followed an iterative testing process to ensure that the requirements were met for each model.

In May 2023, we completed the implementation of the full architecture of our modules and initiated a comprehensive testing phase. Our focus was to evaluate the entire system and ensure smooth interactions between all modules. We conducted rigorous tests to verify the functionality, performance, and compatibility of the text summarizer, text-to-speech model, generated avatar, text-to-emotion, and news genre classification components. The tests encompassed end-to-end scenarios, simulating real-world usage, and assessing the seamless flow of data and communication between the modules. This thorough testing allowed us to identify and address any issues or inconsistencies, ensuring the reliable and cohesive operation of the entire architecture.

5.4 Comparative results from previous work

5.4.1 Text Summarization

Our summarization module was tested on all rouge variants. Our Approach was using Bert along the extractive summarizer, while in the abstractive summarization task, we created the one from scratch.

Model	R1↑	R2↑	RL↑
Extractive Summarizers			
SUMO	41.00	18.40	37.20
TransformerEXT	40.90	18.02	37.17
BERTSUMEXT (Ours)	43.25	20.24	39.63
BERTSUMEXT w/o interval embeddings (Ours)	43.20	20.22	39.59
Abstractive Summarizers			
TransformerABS (Ours)	40.21	17.76	37.09
BERTABS	41.72	19.39	38.76
T5	45.62	25.58	36.53

Table 5.4 Summarization evaluation with other state-of-the-art work

5.4.2 Text to Speech

We compare our model with a parametric (based on LSTM [15]) and a concatenative system [16], both of which are in production.

Model	Mean Opinion Score
Tacotron	3.82 ± 0.085
Parametric	3.69 ± 0.109
Concatenative	4.09 ± 0.119

Table 5.5 Text to Speech evaluation with previous work

5.4.3 Text to Emotion

Our text to emotion module was tested on precision,recall,f1 score metrics . Our Approach Was LSTM layers for the neural network so we compared with other approaches as CNN,SVM.GRUS and RNNS

Method	Precision↑	Recall↑	F1-score↑	Accuracy↑
LSTM (Ours)	0.89	0.90	0.90	0.92
CNN	0.83	0.80	0.81	0.81
Bi-GRU	0.81	0.82	0.82	0.80
RNN	0.89	0.86	0.92	0.94

Table 5.6 Text to Emotion evaluation with previous work

5.4.4 Avatar Generation

For avatar Generation we saw multiple approaches that achieves the lip-sync function and we compared our evaluation metrics with their approaches

Method	LSE-D↓	LSE-C ↑	FID↓
Without Lip-syncing	16.89	2.577	-
Speech2Vid	14.39	1.471	17.96
LipGAN	10.90	3.279	11.91
Wav2Lip (Ours)	9.53	6.41	13.65

Table 5.7 Avatar Generation evaluation with previous work [20]

Chapter 6 Conclusions and Future Work

6.1 Face Challenges

During the development of the Speak News App, several challenges were encountered that required careful consideration and problem-solving. These challenges included:

a) Speech Synthesis Quality: Ensuring high-quality and natural-sounding speech synthesis proved to be a significant challenge. Overcoming issues such as robotic intonation, unnatural pauses, or mispronunciations required fine-tuning of the text-to-speech model, optimization of linguistic processing, and continuous feedback from user testing.

b) Text Preprocessing: Processing and cleaning large volumes of news articles for summarization presented challenges in terms of data collection, formatting, and handling inconsistencies. Dealing with diverse news sources, different writing styles, and varying article structures required robust preprocessing techniques and the development of custom algorithms.

c) User Feedback and Adaptation: Incorporating user feedback effectively into the system and ensuring adaptability to individual preferences and user profiles posed a challenge. Implementing mechanisms for users to provide feedback on the summarization, text-to-speech output, and overall user experience required thoughtful design and constant monitoring.

6.2 Gained Experience

The development of the Speak News App provided valuable learning experiences in several areas, including:

a) Natural Language Processing (NLP): Gained expertise in NLP techniques for text summarization, sentiment analysis, and linguistic processing. This involved understanding and implementing algorithms for extracting key information, sentiment classification, and linguistic feature extraction.

b) Deep Learning Models: Acquired hands-on experience with deep learning models, particularly the Tacotron architecture for text-to-speech synthesis. Explored various neural network architectures, training techniques, and hyperparameter optimization to achieve high-quality speech synthesis.

c) User-Centric Design: Gained insights into user-centric design principles through user feedback sessions and usability testing. Obtained a better understanding of user requirements, preferences, and challenges in the context of news consumption and text-to-speech applications.

6.3 Conclusions

In conclusion, the development of the Speak News App has demonstrated the feasibility and potential of integrating text summarization, sentiment analysis, and text-to-speech synthesis into a unified platform for news consumption. Despite the challenges faced, significant progress has been made in achieving high-quality summarization and natural-sounding speech synthesis.

The user feedback received throughout the development process has been invaluable in refining the system and improving the user experience. The integration of user preferences, sentiment analysis, and personalized text-to-speech synthesis has enhanced the app's usability and relevance for a wide range of users.

6.4 Future Work

There are several avenues for future work and enhancements for the Speak News App, including:

a) Expansion of News Sources: Incorporating a wider range of news sources and languages to increase the app's coverage and accessibility to a more diverse user base.

b) Multimodal Integration: Exploring the integration of additional modalities, such as images or video summaries, to enhance the news consumption experience and cater to different user preferences.

c) Continuous Model Improvement: Continuously refining and fine-tuning the text summarization, sentiment analysis, and text-to-speech models using user feedback and advanced machine learning techniques to enhance accuracy, naturalness, and personalization.

d) User Profiles and Recommendations: Developing user profiles based on individual preferences, interests, and feedback to provide personalized news summaries and tailored text-to-speech outputs. Incorporating recommendation systems to suggest relevant news articles and enhance the user experience.

e) Accessibility Features: Incorporating accessibility features, such as closed captioning, voice commands, and integration with screen readers, to ensure inclusivity and usability for users with disabilities.

f) Performance Optimization: Optimizing the performance of the app, including reducing latency, enhancing scalability, and improving resource efficiency to ensure smooth and efficient user experience.

These future directions will further enhance the Speak News App's capabilities.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.
[2] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention based neural machine translation. arXiv preprint arXiv:1508.04025, 2015.
[3] Liu, Y. (2019). Fine-tune BERT for extractive summarization. *arXiv preprint arXiv:1903.10318*.
[4] Liu, Y., & Lapata, M. (2019). Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.
[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.
[6] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410, 2016.
[7] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. ˙ Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, pages 280–290, Berlin, Germany
[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.
[9] Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In Proceedings of the ACL Conference.
[10] Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In Proceedings of the NAACL Conference.
[11] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450
[12] Lin, Chin-Yew. (2004). ROUGE: A Package for Automatic Evaluation of summaries. Proceedings of the ACL Workshop: Text Summarization Braches Out 2004. 10.
[13] Wang, Z., Weiss, R. J., & Kingsbury, D. (2017). Tacotron: Towards end-to-end speech synthesis. In Proceedings of the 34th International Conference on Machine Learning (pp. 5066-5074).
[14] Xavi Gonzalvo, Siamak Tazari, Chun-an Chan, Markus Becker, Alexander Gutkin, and Hanna Silen. Recent advances in Google real-time HMM-driven unit selection synthesizer. In Proc. Interspeech, pp. 2238–2242, 2016.
[15] Heiga Zen, Yannis Agiomyrgiannakis, Niels Egberts, Fergus Henderson, and Przemysław Szczepaniak. Fast, compact, and high-quality LSTM-RNN based statistical parametric speech synthesizers for mobile devices. Proceedings Interspeech, 2016.
[16] Valerio Velardo. “How to Extract Audio Features” Publisher, uploaded by Youtube, Jul 16, 2020,
[17] Su, M. H., Wu, C. H., Huang, K. Y., & Hong, Q. B. (2018, May). LSTM-based text emotion recognition using semantic and emotional word vectors. In 2018 First Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia) (pp. 1-6). IEEE.
[18] Huang, F., Li, X., Yuan, C., Zhang, S., Zhang, J., & Qiao, S. (2021). Attention-emotion-enhanced convolutional LSTM for sentiment analysis. IEEE transactions on neural networks and learning systems, 33(9), 4332-4345.
[19] T. Afouras, J. S. Chung, A. Senior, O. Vinyals, A. Zisserman. The Oxford-BBC Lip Reading Sentences 2 (LRS2) Dataset. Lip Reading Sentences 2 (LRS2) dataset. (n.d.). https://www.robots.ox.ac.uk/~vgg/data/lip_reading/lrs2.html
[20] Prajwal, K. R., Mukhopadhyay, R., Namboodiri, V. P., & Jawahar, C. V. (2020, October). A lip sync expert is all you need for a speech to lip generation in the wild. In Proceedings of the 28th ACM International Conference on Multimedia (pp. 484-492).
[21] Chung, J. S., & Zisserman, A. (2017). Out of time: automated lip sync in the wild. In *Computer Vision–ACCV 2016 Workshops: ACCV 2016 International Workshops, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part II 13* (pp. 251-263). Springer International Publishing.
[22] Trivedi, C. (2020, August 31). *Deepfakes AI-improved lip sync animations with wav2lip*. Medium. https://medium.com/deepgamingai/deepfakes-ai-improved-lip-sync-animations-with-wav2lip-b5d4f590dcf
[23] Chung, J. S., Jamaludin, A., & Zisserman, A. (2017). You said that?. *arXiv preprint arXiv:1705.02966.*

Appendix A: Development Platforms and tools

A.1 Hardware Platforms

1) Training was done with different GPUS on the Laptops of the team members which are:
    a. Nvidia GTX 1060
    b. Nvidia GTX 1650
    c. Nvidia RTX 3050 Ti
    d. Kaggle p100
    e. Our computer engineering department GPUs
2) MacBook pro M2 for developing and deploying the mobile application.
3) Samsung A022 for testing the mobile application.

A.2 Software Platforms

1) Python: Python is a popular programming language used for machine learning due to its extensive library ecosystem and ease of use.

   Anaconda: A Python distribution that includes many pre-installed libraries for data science and machine learning.

2) TensorFlow: An open-source library developed by Google for numerical computation and large-scale machine learning. Keras: A high-level neural networks API that runs on top of TensorFlow, making it easier to build.

3) PyTorch: An open-source machine learning library developed by Facebook's AI Research Lab that provides dynamic computational graphs and efficient GPU acceleration. And train deep learning models.

4) Scikit-learn: A machine learning library built on NumPy, SciPy, and Matplotlib that provides a wide range of algorithms and tools for classification, regression, clustering, and dimensionality reduction.

5) Jupyter Notebook: An interactive web-based tool that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It's commonly used for exploratory data analysis and prototyping machine learning models.

6) Pandas: A powerful data manipulation and analysis library that provides data structures and functions to efficiently handle structured data.

7) NumPy: A fundamental package for scientific computing in Python that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

8) Matplotlib: A plotting library for creating static, animated, and interactive visualizations in Python.

9) XGBoost: An optimized gradient boosting library that provides a highly efficient implementation of the gradient boosting framework.

10) OpenCV: An open-source computer vision library that provides a comprehensive set of functions for image and video processing.

11) NLTK is Python library used with human language data. It offers many text processing techniques such as tokenization, stemming, tagging and parsing.

12) The MIT Language Modeling (MITLM) toolkit is a set of tools designed for the efficient estimation of statistical n-gram language models.

Appendix B: Use cases

1) **Accessibility for Visually Impaired Users**: By utilizing the text-to-speech model and generated avatar, your project can benefit visually impaired users by converting written news articles into spoken content. The news summarizer helps in condensing the information, allowing visually impaired individuals to access news quickly and conveniently.

2) **In-Car News Service**: Integrating your project into vehicles or navigation systems allows drivers to stay informed about the latest news while on the road. The news summarizer can select and condense relevant articles, and the text-to-speech model and generated avatar can deliver the news updates hands-free, ensuring driver safety.

3) **Podcast or Radio Show Enhancement**: Podcast hosts or radio show producers can utilize your project to convert written news articles into spoken content. The news summarizer helps in selecting the most relevant news items, and the text-to-speech model and generated avatar can narrate the news, enriching the show's content and reducing the need for human voiceover.

4) **Educational Applications**: Your project can be used in educational settings to promote news literacy and engagement. Students can receive summarized news articles, listen to the narrations, and discuss current events. The combination of summarization, text-to-speech, and generated avatars adds an interactive and engaging element to learning about current affairs.

5) **New Way for Youth News Consumption**: Our application offers a fresh and captivating approach to news consumption specifically designed for youth. By integrating a news summarizer, text-to-speech model, and generated avatar, we provide a unique platform for young individuals to engage with news content. The summarization feature condenses news articles into concise summaries, enabling quick and efficient information absorption. The text-to-

speech model converts these summaries into audio format, catering to the growing preference for audio content among youth. Additionally, the generated avatar adds a personalized touch by narrating the news with an interactive and engaging voice. With our application, youth can seamlessly listen to news updates, stay informed about current events, and foster their news literacy skills in an innovative and captivating manner. We strive to redefine how young individuals consume news, offering them a novel and tailored experience that aligns with their preferences.

Appendix C: User Guide

1. First of all, let's start with the user interface that the user experiences it's a feed of news that is fetched from CNN website. The news is from various genres, sports, business, political, entertainment, Science.

2. Then the news is classified and divided according to each genre and the user could now access each genre separately

3. We also have the feature of searching which enables the user to search for his desired topic.

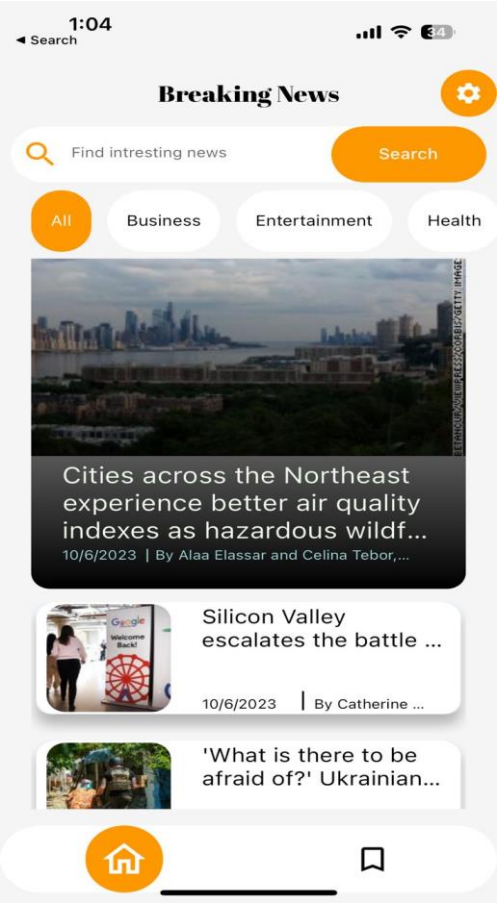


Figure 0.3 Home Page of Our application

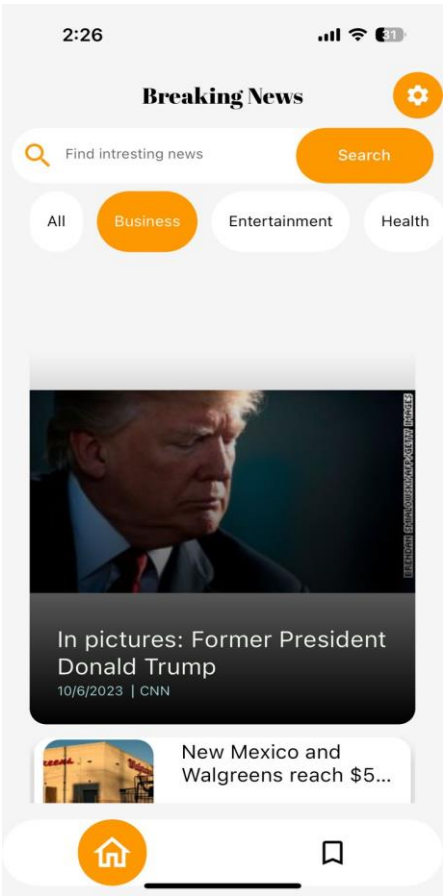


Figure 0.1 Genres of news divided separately.

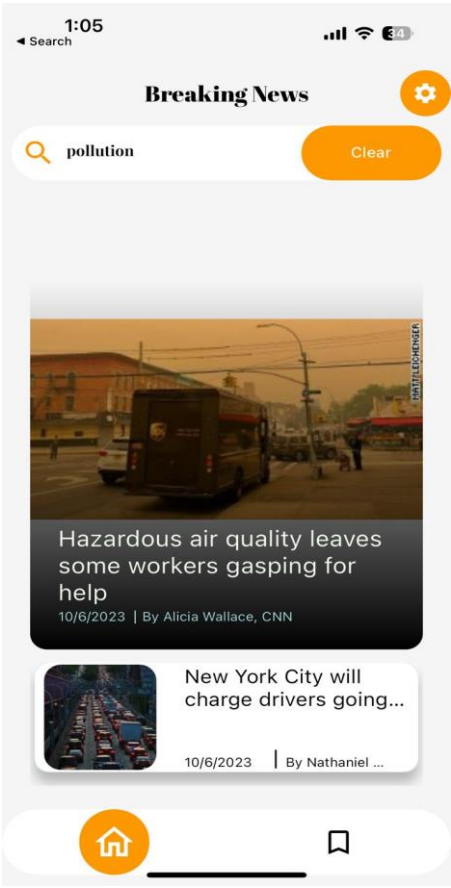
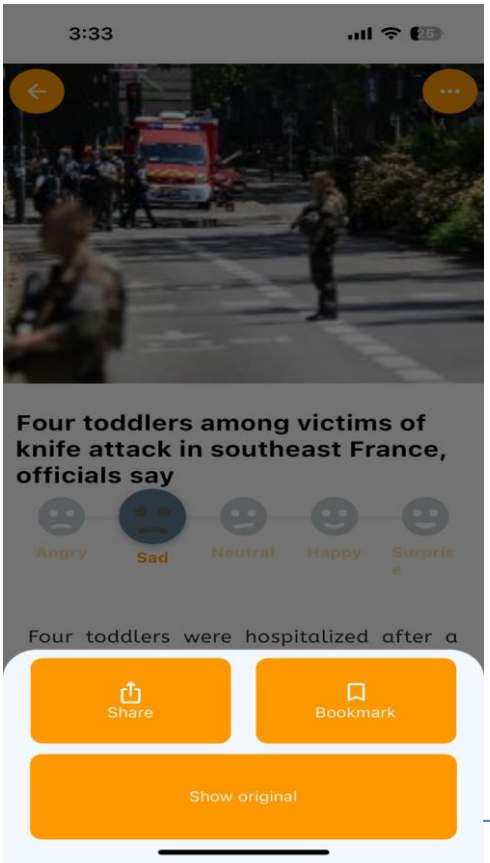


Figure 0.2 Searching for desired topic.

4. User can bookmark his favorite news so he can access them later.
5. User can use between extractive and abstractive summarization approaches, and we give him hints on what basis he could to choose between them.
6. For extractive summarization we allow user to choose the compression ratio for text.



Press on this button to access the feature of bookmark.

Bookmark this article

Figure 0.5 How to bookmark your article

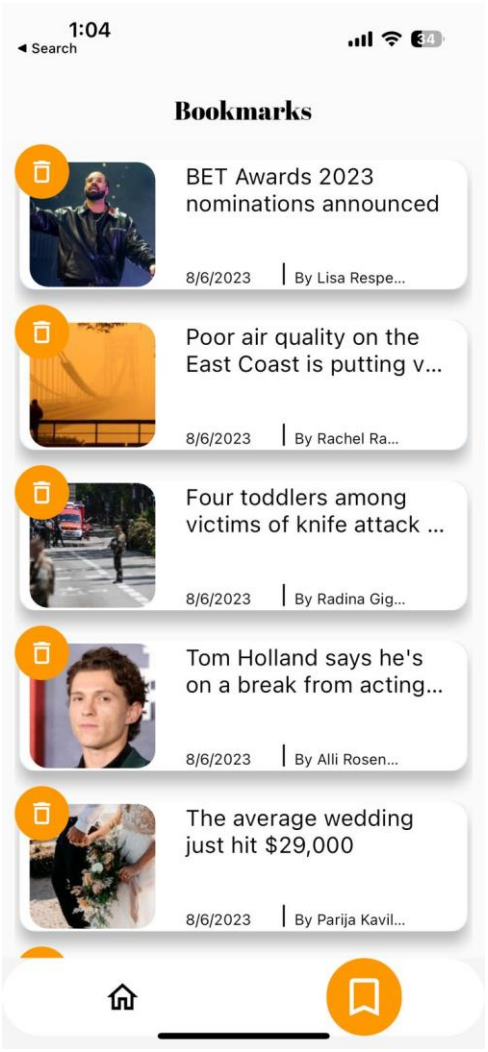


Figure 0.4 Bookmarks Page of the User

Figure 0.8 setting Icon

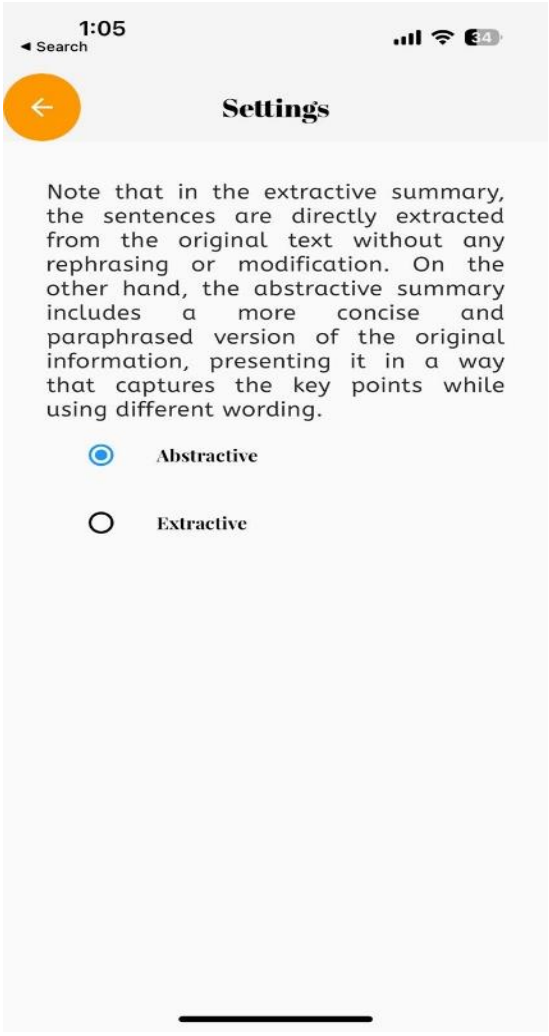


Figure 0.7 Choose between extractive and abstractive summarization.

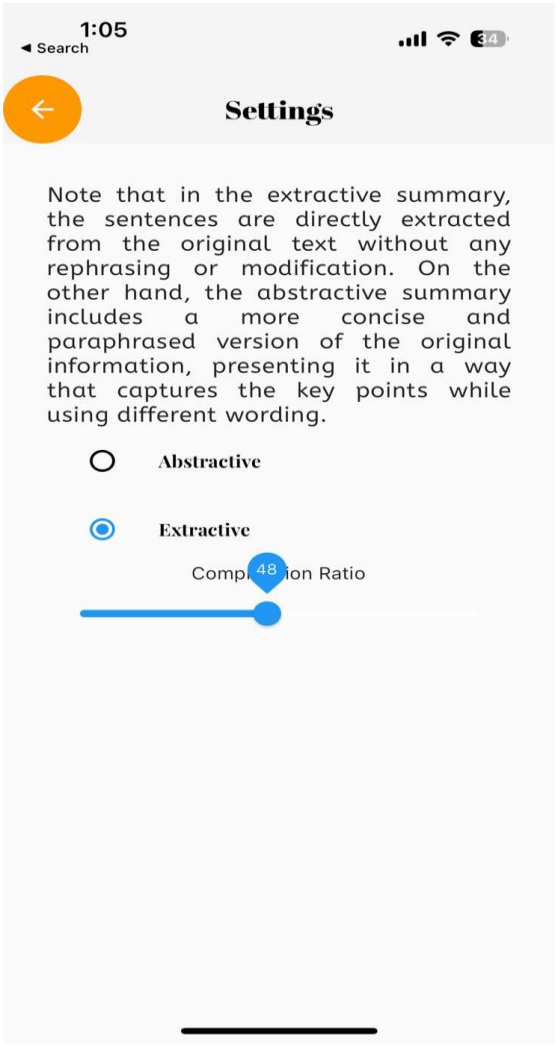


Figure 0.6 Choose compression ratio for extractive summarization.

7. User can now generate audio and avatar video from his chosen article.

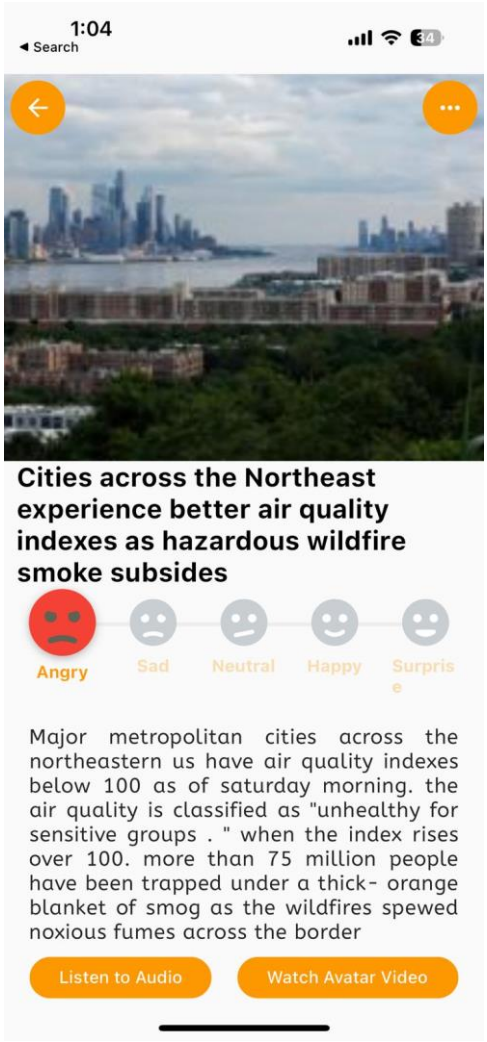


Figure 0.11 choosing between generating audio or video.

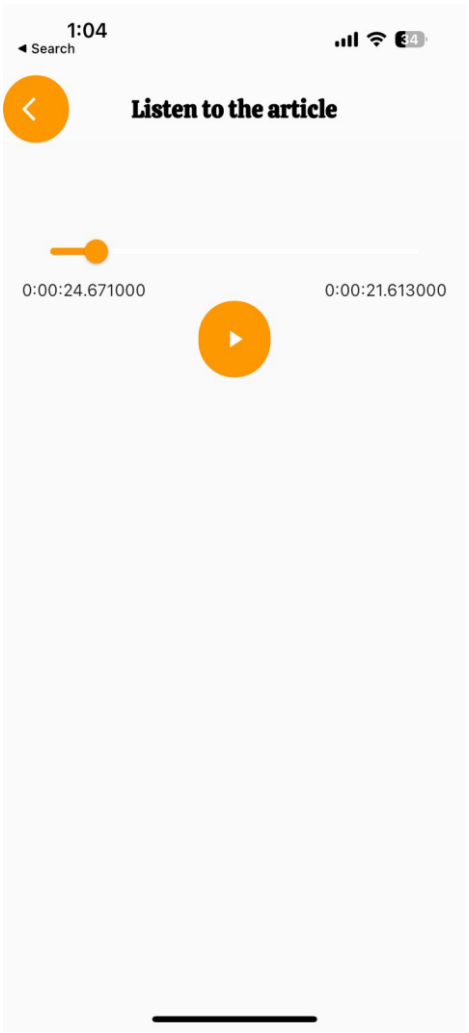


Figure 0.10 Generated audio file for the article.

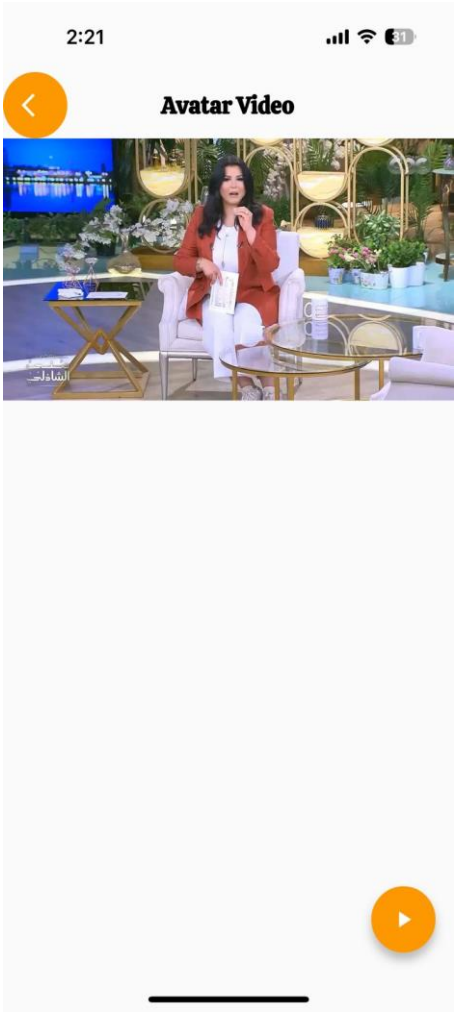


Figure 0.9 Generated avatar for the article.

Appendix D: Feasibility study

Executive Summary:

This feasibility study examines the viability of developing a comprehensive news consumption solution that includes a news summarizer, text-to-speech model, and generated avatar for news narration. The study assesses technical feasibility, financial viability, market potential, legal considerations, and environmental impacts. Based on the findings, the project is deemed technically feasible, financially viable, holds substantial market potential, complies with legal requirements, and has minimal environmental impact. Consequently, it is recommended to proceed with the development and implementation of the project.

Technical Feasibility:

The technical feasibility evaluation focuses on assessing the availability and suitability of software, tools, and technologies required for the project. Python will serve as the primary programming language, and various libraries and frameworks such as TensorFlow, PyTorch, Keras, and OpenCV will be utilized for machine learning tasks, natural language processing, and computer vision. These widely adopted technologies provide a robust foundation for developing the news summarizer, text-to-speech model, and generated avatar functionalities. Integration can be achieved through established APIs, enabling seamless communication between the components.

Financial Feasibility:

The financial feasibility analysis examines the costs associated with the project's development, deployment, and maintenance. Factors such as hardware and software resources, skilled personnel, operational expenses, and potential revenue streams are considered. Initial investments will cover acquiring suitable hardware infrastructure and hiring a competent team of developers and data scientists. Ongoing operational costs include hosting, maintenance, and continuous model updates. Revenue streams may come from licensing the technology to news platforms, offering premium features, or partnerships with media organizations. Through careful financial planning, cost management, and revenue generation strategies, the project demonstrates long-term financial viability.

Market Feasibility:

Market feasibility assesses the project's potential success in the target market. Extensive market research reveals a growing demand for personalized and accessible news consumption, particularly among youth and individuals seeking convenient and engaging news experiences. By combining a news summarizer, text-to-speech model, and generated avatar, the project caters to this demand by providing a unique, interactive, and tailored news consumption solution. Differentiation in the market is achieved through the ability to condense news articles into concise summaries, convert them into audio format, and narrate them with a generated avatar. This innovative approach positions the project favorably to capture a significant market share and establish partnerships with news platforms and content providers.

Legal Feasibility:

Legal feasibility ensures compliance with intellectual property rights, data privacy regulations, and any other legal requirements. Proper attribution and licensing of sources will be implemented to respect copyright laws. To safeguard user data, stringent data privacy measures will be adopted, including anonymization, and obtaining user consent for data processing. Compliance with applicable laws and regulations will be overseen by legal experts to mitigate any legal risks.

Environmental Feasibility:

Environmental feasibility evaluates the project's impact on the environment throughout its lifecycle. As a software-based solution, the project has minimal direct environmental impact. However, efforts will be made to optimize resource utilization and energy efficiency during development and deployment. Cloud infrastructure will be selected with sustainability considerations in mind, prioritizing providers that use renewable energy sources and promote energy-efficient data centers.

WEE Water Engineering and Environment

STE Structural Engineering

PPC Petro Chemical Engineering

MDE Mechanical Design Engineering

CEM Construction Engineering and Management

CCE Communication and Computer Engineering

AET Architectural Engineering and Technology

Sponsor

Sponsor Logo