CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, ISLAMABAD

Abstractive Text Summarization in Urdu Using Multiple Encoders and a Single Decoder with Semantic Extractor

by

Wajiha Fatima

A thesis submitted in partial fulfillment for the degree of Master of Science

in the

Faculty of Computing
Department of Computer Science

2025

First of all, I am dedicating this thesis to my maternal mother, Shamim Akhtar, although she is no longer of this world, who loved me and taught me the value of life. May you find peace and happiness in paradise. I also dedicate this thesis to my parents, who have always trusted me and supported me through every challenge. Furthermore, my thesis honors all those who have helped me through good times and bad.

CERTIFICATE OF APPROVAL

Abstractive Text Summarization in Urdu Using Multiple Encoders and a Single Decoder with Semantic Extractor

by

Wajiha Fatima

(MCS223008)

THESIS EXAMINING COMMITTEE

S. No.	Examiner	Name	Organization
(a)	External Examiner	Dr. Hasan Mujtaba	FAST-NU, Islamabad
(b)	Internal Examiner	Dr. Nayyer Masood	CUST, Islamabad
(c)	Supervisor	Dr. Syed Saqib Raza Rizvi	CUST, Islamabad

Dr. Syed Saqib Raza Rizvi
Thesis Supervisor
February, 2025

Dr. Abdul Basit Siddique
Head
Dept. of Computer Science
February, 2025

Dr. M. Abdul Qadir
Dean
Faculty of Computing
February, 2025

Author's Declaration

I, **Wajiha Fatima** hereby state that my MS thesis titled "**Abstractive Text Summarization in Urdu Using Multiple Encoders and a Single Decoder with Semantic Extractor**" is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/abroad.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw my MS Degree.

(Wajiha Fatima)

Registration No: MCS223008

Plagiarism Undertaking

I solemnly declare that research work presented in this thesis titled "**Abstractive Text Summarization in Urdu Using Multiple Encoders and a Single Decoder with Semantic Extractor**" is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Capital University of Science and Technology towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS Degree, the University reserves the right to withdraw/revoke my MS degree and that HEC and the University have the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized work.

(Wajiha Fatima)
Registration No: MCS223008

Acknowledgement

First of all, I want to express my gratitude to Almighty Allah for giving me the strength to complete this thesis. I sincerely thank my supervisor, Dr. Syed Saqib Raza, for being so supportive, expressing such strong confidence in me, and remaining very kind to me. Throughout my thesis journey, I sincerely appreciate your guidance, priceless advice, and unwavering support.

Furthermore, I would want to thank my family and friends for all their invaluable support and encouragement during this task. Finally, I would want to express my gratitude to my seniors and classmates for lending me their time and efforts in order to accomplish this task. I extremely grateful to all of them.

(Wajiha Fatima)

Abstract

In the field of Natural Language Processing (NLP), the task of text summarization holds great importance due to its ability to comprehend textual content and generate concise summaries. Text summarization can be either extractive or abstractive in nature. However, the domain of abstractive summarization for the Urdu language remains largely unexplored. This study specifically focuses on designing and implementing an abstractive text summarization model for Urdu that effectively captures the semantic meanings of the original text. The primary objective of this research is to increase the readability and understanding of the original content by producing a concise and thorough overview of the Urdu language that includes new words and expressions. The proposed approach generates abstractive summaries by utilizing a multi-layer encoder architecture along with a pre-trained multilingual mBART model. The paper shows positive outcomes for abstractive text summarization in Urdu, achieving a *BERTScore* of 90%, a *BLEU* score of 43%, and *ROUGE* scores of 80.3% for ROUGE-1, 74.3% for ROUGE-2, and 80.3% for ROUGE-L. The study demonstrates the use of a robust multi-layer encoder-decoder architecture to produce concise, coherent summaries in the Urdu language, enhancing readability and comprehension and improving overall summarization quality.

Contents

List of Figures

List of Tables

Abbreviations

Al	Artificial Intelligence
BOW	Bag of Words
CNN	Convolutional Neural Network
DUC	Document Understanding Conference
ICASSP	International Conference on Acoustics, Speech, and Signal Processing
ICML	International Conference on Machine Learning
LDA	Latent Dirichlet Allocation
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
Tf–Idf	Term Frequency-Inverse Document Frequency

Symbols

a	Attention weight
$E(w)$	Embedding vector of a word/token w
$FFN(x)$	Feedforward neural network function applied to input x
Wi	Weight of sentence or token i
ct	Context vector in attention mechanisms
t	Attention weights at time step t
Q, K, V	Query, Key, and Value matrices in attention mechanisms
d	Embedding space size

Chapter 1

Introduction

1.1 Introduction

The ability of machines to carry out operations that normally require human intelligence, such as learning, reasoning, problem solving, and language comprehension, is known as artificial intelligence (AI) [1]. The ability of machines to comprehend and communicate with human language is known as natural language processing (NLP); machine learning (ML), which includes supervised, unsupervised, and reinforcement learning; and deep learning, a subset of ML that processes enormous volumes of data using intricate neural networks [2]. Additionally, robotics integrates AI into physical machines for autonomous or semi-autonomous operations, while expert systems use rules and a knowledge base to mimic human decision-making [3]. Lastly, computer vision enables computers to decipher visual information, which makes it easier to use applications like autonomous car obstacle detection and facial recognition. Collectively, these AI components are revolutionizing industries, improving technological interfaces, and holding out the promise of major breakthroughs in a number of fields [4].

Figure 1.1 illustrates the general phases of NLP by dividing the examination of an input sentence into discrete linguistic processing phases; this figure illustrates the Natural Language Understanding (NLU) process. It explains how meaning and contextual relevance are extracted from raw linguistic data (an input sentence)

through a series of analyses[5].The language processing pipeline begins with the raw text, which is usually a single sentence like "The cat is sitting on the mat." The analysis and comprehension of the root form, prefixes, suffixes, and grammatical functions of individual words within a sentence constitute the process of morphological processing. Part-of-speech (POS) tagging, tokenization, and lemmatization are used to break the input text up into words, change the words to their base form, and determine grammatical categories. Words like "cat," "is," "sitting," "on," "the," and "mat" are examples of output.

Syntax analysis, a phase in grammar processing, involves examining a sentence's grammatical structure to ensure compliance with syntax rules and identifying the relationships between words. It includes dependency parsing to ascertain how terms in the phrase, such as subjects, objects, and modifiers, rely on one another and parsing a sentence to produce a syntactic tree. Building a parse tree serves as an illustration of this. The process of semantic analysis assigns meaning to words and determines their context by interpreting phrase meanings according to semantic norms.

Word Sense Disambiguation (WSD) is used to assign roles such as actor, action, and object, and semantic role labeling is used to ascertain the right meaning of words having multiple senses. "Cat" is the agent, "sitting" is the action, and "mat" is the object, for instance.Pragmatic analysis is the process of analyzing a sentence's meaning based on situational context and outside information, as well as comprehending the sentence's context and practical consequences. It entails addressing ambiguities, evaluating implicit meaning, and resolving references. In a tale, for instance, the system may decide that "the mat" denotes a certain spot in the room and "the cat" is a pet.Machine translation, sentiment analysis, and conversational AI applications can all benefit from the process's output, which is a sentence-structured representation that contains both explicit and implicit meanings.

Virtual assistants such as Siri, Alexa, and Google Assistant are powered by NLP, which enables users to manage devices, create reminders, and retrieve information using voice requests. NLP is used to translate text between languages by services

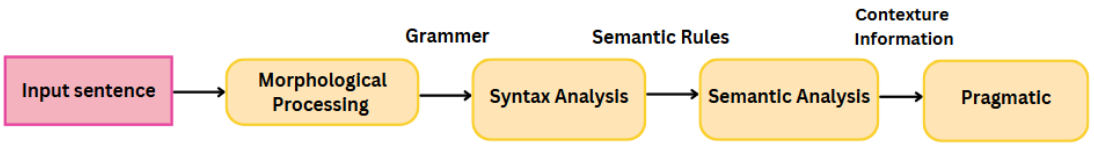

FIGURE 1.1: General Phases of Natural Language Processing.

like Google Translate, which facilitates interlanguage communication. NLP is a tool used by businesses to examine comments from online forums, social media, and reviews in order to determine how the public feels about certain events, companies, or items [6]. News articles can be sorted by topic or spam in emails thanks to NLP algorithms' ability to categorize text. NLP is used in systems such as IBM Watson to interpret context and retrieve pertinent information in order to offer accurate replies to user inquiries [7]. Search engines use NLP to better comprehend user queries and return pertinent documents or web pages based on semantic meaning. GPT-3 and other NLP models may produce text that appears human for a variety of uses, such as chatbots, automated news stories, and creative writing. Applications like customer service, transcription services, and accessibility tools are made possible by NLP, which allows systems to convert spoken language into text [6]. In order to help readers efficiently assimilate information, NLP techniques are utilized to produce succinct summaries of extensive public Chatbots powered by NLP are used by businesses to improve customer engagement and provide prompt answers to user questions [7].

NLP-driven chatbots are used by businesses for customer support, offering prompt answers to user questions and enhancing user engagement [4]. For applications like search engine autocomplete and messaging apps, NLP is crucial for anticipating the next word or phrase in a conversation. Grammarly and other tools employ NLP to check writing for grammatical errors, style changes, and readability issues. NLP helps in information extraction and organization by recognizing and categorizing textual items, such as names, organizations, and locations [5]. Based on user input, NLP may generate dynamic plots and character interactions for video games and other interactive applications [4]. NLP is used by businesses to track trends and brand mentions on social media, enabling them to react instantly to consumer input. By identifying similarities in texts, NLP techniques

can help ensure academic integrity by spotting possible plagiarism [5]. Businesses can design user-interest-based marketing strategies by using NLP to assess consumer behavior and preferences [6]. By using NLP to evaluate applications and cover letters, employers may find qualified applicants more quickly [6]. NLP helps lawyers evaluate legal papers by highlighting important terms, summarizing cases, and extracting pertinent data [6].

Machines are challenged by the diverse meanings and context-dependent features of human language; also, obtaining high-quality labeled data for NLP models can be difficult, and biases in training data may deliberately spread [7].

In NLP, text summarization is a difficult process [7]. To make reading and searching information from multiple articles easier, it uses condensed versions without compromising significance [8]. Due to the Internet's fast expansion. For the past two decades, the Internet has been a fantastic source for news, articles, and book reviews, and this tendency is only going to get stronger [7]. Textual data is rapidly growing as a result of the overwhelming amount of data. Internet users use search queries to find information [8]. It is nevertheless time-consuming and demanding for the user to visit several websites in order to find the information they require [7]. In order to avoid this headache, manage this massive amount [8]. One technique developed to extract information from an article with the least amount of data and in the shortest amount of time is text summarization [8]. A General Text Summarizer model, as shown in Figure 1.2, is described [9].

A text summarizing workflow is depicted in figure 1.2, where text data is processed to identify important sentences and provide a summary[10]. The term "data source" describes the location of textual data, which may be databases, file systems, or online resources such as research papers or news articles. Among the methods are APIs, web scraping tools, and database queries. Based on input queries or requirements, the document fetch method uses file I/O operations, API calls for online or cloud-based retrieval, and document parsers for certain formats to obtain pertinent documents from a data source. A technique called Tokenize Sentence divides a text into separate sentences in order to get it ready for examination. It handles any misunderstandings brought on by acronyms and works

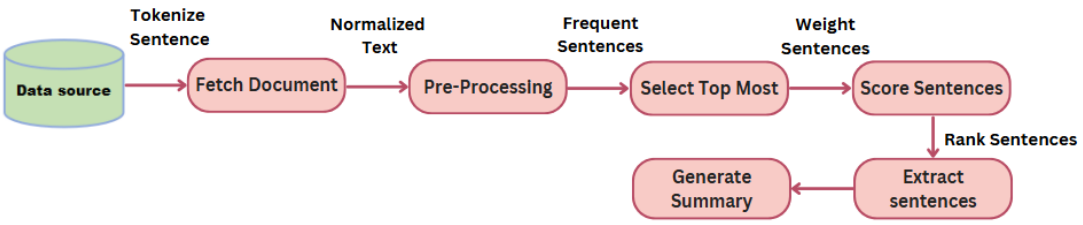

FIGURE 1.2: A General Text Summarizer Model.

with multilingual content by segmenting the text using NLP libraries like NLTK and SpaCy. Preprocessing ensures consistency and eliminates noise by cleaning and normalizing text for analysis. Lowercasing, eliminating stop words, stemming/lemmatization, eliminating special characters, and managing synonyms are all part of it. Regular expressions are used to eliminate particular patterns, and text preparation tools such as NLTK, SpaCy, or Stanford CoreNLP are employed. This procedure aids in eliminating punctuation, numerals, symbols, context, and grammatical structure. According to the frequency or significance of words, candidate sentences can be found using the Select Top Most (Frequent Sentences) step. Among the methods are keyword extraction, TF-IDF, and term frequency (TF). Key keywords are found using methods like RAKE or word embeddings, whereas common words are given less weight. Sentences are given significance scores by the Score Sentences method according to how pertinent they are to the topic. Semantic similarity, position-based weighting, N-gram models, the Bag of Words (BoW), the TextRank algorithm, and the PageRank algorithm are some of the methods. While PageRank ranks sentences, semantic similarity gauges how similar phrases are to one another, BoW is a frequency-based scoring system, and TextRank generates a network of sentences.

The summary uses ranking algorithms to order the phrases according to their ratings, diversity-based selection to guarantee that a wide range of subjects are included in the summary, and redundancy elimination to avoid too many similar lines. The technology ensures logical flow and accessible information by merging collected sentences to create a cohesive summary. Based on subjects, user preferences, or relevance, it provides customization choices and modifies the compression ratio to maintain the desired summary length. Among the tools are

post-processing methods for grammar or style enhancement and basic sentence concatenation. Text summarizing techniques can be categorized into abstractive and extractive approaches based on the type of summary that is generated [11]. A general text approach, as shown in Figure 1.3, is described [12]. The input layer, the output layer, and the summarization's goal are the three primary criteria used in this figure to categorize text summarizing approaches. The input layer is used to categorize summarization techniques. News pieces, scholarly papers, and online pages can all benefit from a single document summary, which reduces text while maintaining key concepts. Multi-document summarization handles redundant, conflicting, and complimentary information from different sources while synthesizing information from numerous documents. This method is employed in a corporate context to summarize several reports or to produce relevant news pieces. Avoiding redundancy, maintaining coherence, and combining data from several sources are challenges.

Abstractive and extractive summarization are the two categories of summarizing that are distinguished by the output layer. Similar to a human summary, abstractive summarization entails crafting new phrases that capture the essence of the source material. Among the methods are transformer models, neural networks, and Seq2Seq models. This is helpful for writing natural and concise summaries for articles, tales, or reports. With extractive summary, the most significant sentences or phrases are chosen from the source material without creating new material. Among the methods are sentence scoring, clustering, TextRank and PageRank algorithms, and TF-IDF. This is helpful for technical reports or legal papers when maintaining the original language is essential. Summarization is carried out for certain objectives. Answering research inquiries or summarizing papers for certain keywords are examples of how query-based summarization is designed to address particular queries or concentrate on particular subjects within a text. Domain-Specific Summarization is tailored to a particular subject or sector, such as science, medicine, law, or finance. Specialized processing and topic expertise are frequently needed for this kind of summary. A medical summarizing tool, for instance, might distill a research paper into conclusions that are applicable to clinical practice. Generating overviews of books or essays or summarizing broad news stories are

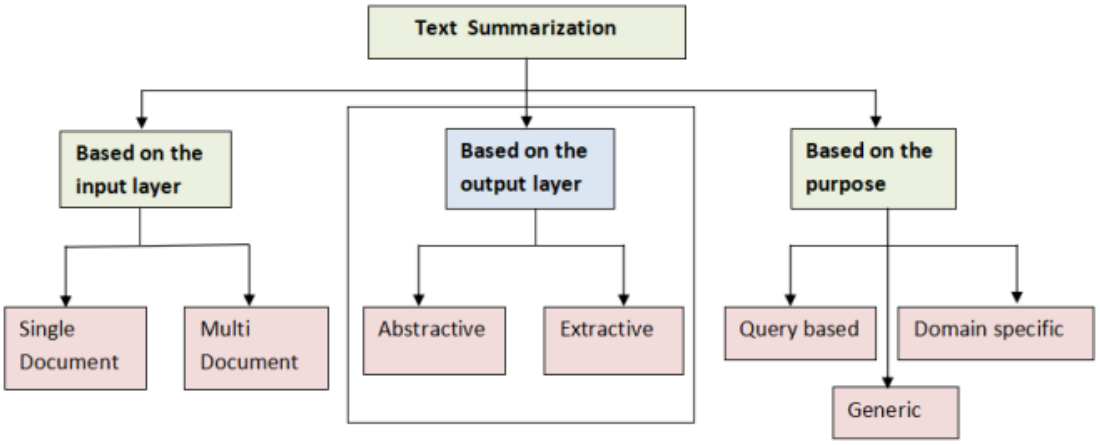

FIGURE 1.3: Text Summarization Approaches.

examples of generic summarization, which offers a general summary without concentrating on any particular subject or question. This kind of synopsis emphasizes the incident, its importance, and its conclusion.

Instead of generating new terms or rephrasing content, extractive summarization focuses on identifying and highlighting the most important sentences or phrases directly from the original text. This approach utilizes various linguistic or statistical criteria to determine the significance of individual components within the text. Extractive summarization essentially selects key sentences that represent the main ideas, without altering the original wording or structure [13]. In contrast to abstractive summarization, which creates summaries using newly generated phrases or sentences, extractive summarization is concerned with the careful extraction of relevant content from the source material. A typical architecture of an extractive text summarization system, as illustrated in Figure 1.4, involves several stages, such as text preprocessing, feature extraction, sentence ranking, and selection, to ensure that the most representative and meaningful sentences are retained [14].

The phases and components of extractive text summarization are depicted in figure 1.4. Key sentences or phrases are extracted from the input content, which might be a long document, article, or body of text.Preprocessing is an essential stage in summarization, which entails dividing the input text into smaller chunks, such as words or sentences, eliminating stop words, and standardizing it using techniques

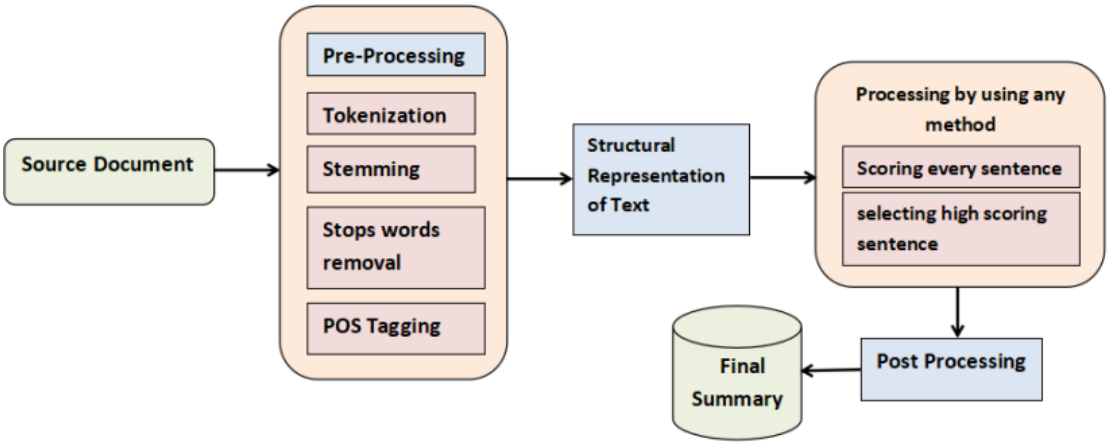

FIGURE 1.4: Framework of an Extractive Text Summarization System.

like lemmatization, stemming, or lowercasing to reduce words to their most basic forms.

After preprocessing, the algorithm identifies sentence properties such as location, length, and phrase frequency before extracting features from the text. Because sentences can serve as indicators of the start or finish of a text and their weight, these characteristics aid in determining the significance of sentences. Using algorithms like TF-IDF, sentence scoring determines a word's significance in a text by allocating a score based on attributes that have been retrieved. Significant words with high TF-IDF scores have greater relevance. Similarity between words is represented by edges in graph-based algorithms such as TextRank, which regard sentences as nodes in a graph.

The phrases that receive the highest ratings in extractive summary are the most significant and pertinent to the main concepts of the text. The last stage in the summary generating process is to concatenate the top-ranked sentences in a logical manner while preserving the primary ideas of the original text. Readers may immediately understand the major elements of the content without changing its language because to the output summary's conciseness and emphasis on maintaining the original phrase. Applications in extractive summarization include legal research, public opinion analysis, search engine optimization, meeting summaries, online course material reduction, patient data analysis, and financial news and

trends are included in this summary [15]. Search engine optimization, social media analysis, and document summarization for improved comprehension and recall are also covered [16]. When extractive summarizing algorithms encounter difficulties with informal vocabulary, idioms, or intricate sentence patterns, the result may be summaries that are inconsistent or fail to capture the nuances of the source material [17]. Algorithms may unintentionally introduce bias by overemphasizing some parts of the text while disregarding others since they may constantly favor particular phrase types. This may distort the synopsis and present the original information incorrectly [18]. Long texts with primary points dispersed throughout multiple sections can be difficult for extractive summarization to efficiently summarize without leaving out important elements. Repetitive use of similar sentences frequently results in redundancy, which can lengthen the summary and obscure its meaning [18]. Extractive algorithms may fail to recognize important but small differences when summarizing domain-specific materials, such as legal or medical records, if they do not possess unique contextual knowledge [19].

Abstractive summarizing is defined as a concise and logical summary that may include material that has been condensed or altered but was not included in the original text [20]. This method goes beyond simply selecting and extracting sentences from the source text to produce fresh, human-like summaries by understanding and interpreting it [21]. Abstractive summary techniques, as opposed to extractive methods, often employ deep learning models and natural language processing to generate summaries that more imaginatively and flexibly capture the essence of the given content [22]. Nonetheless, problems with abstractive summarizing methods are still common and include maintaining coherence, factual accuracy, and controlling out-of-domain information [23].

Figure 1.5 illustrates the general architecture of abstractive text summarization, [24]. The goal of abstractive summary, as opposed to extractive summarization, is to produce new phrases that express the same meaning. The input material can be any long document, article, or body of text. Preprocessing is the process of dividing the input text into smaller units for analysis, such words or sentences, and then compiling the text into a standard format for analysis, like lemmatization,

stemming, or lowercasing. A dense vector representation is produced by preprocessing the text and then running it through an encoder. Using methods like Word2Vec, GloVe, or contextual embeddings from models like BERT or mBART, word embeddings turn each word into a vector. A contextual representation then interprets the full input sequence to determine the meaning of the text.When creating a summary, an attention technique is employed to concentrate on various sections of the supplied text. For every word in the output summary, this aids in identifying the most pertinent words or phrases. The summary is produced by the decoder utilizing output tokens and context to forecast the subsequent word. The output is subjected to a softmax function, which generates a probability distribution over the vocabulary, enabling the model to choose the most likely subsequent word.

Abstractive summarization is a method that generates new phrases that capture the essence of the original text, rather than extracting sentences directly from the source text. This approach involves paraphrasing, restructuring sentences, and sometimes introducing novel words and expressions while preserving the intended meaning. The core of abstractive summarization is a sequence-to-sequence (Seq2Seq) model, typically incorporating an encoder-decoder architecture. The decoder learns to rephrase information from the input text, and the model is optimized to reduce discrepancies between generated and high-quality reference summaries. This optimization is driven by loss functions like cross-entropy loss, which penalizes incorrect predictions and enhances the model's ability to generate fluent, accurate summaries over time.

Training on a sizable and varied dataset greatly improves the model's capacity to retain contextual relevance, grammatical accuracy, and logical coherence while capturing key concepts. As the model is exposed to a wide range of topics, writing styles, and textual structures, it develops the ability to provide summaries that are not only fluid but also relevant and appropriate for the context. Because of its lengthy training, the model is better able to generalize across diverse domains and adapt to new kinds of material. In order to evaluate abstractive summarization models, researchers use well-known evaluation metrics as BERTScore, BLEU

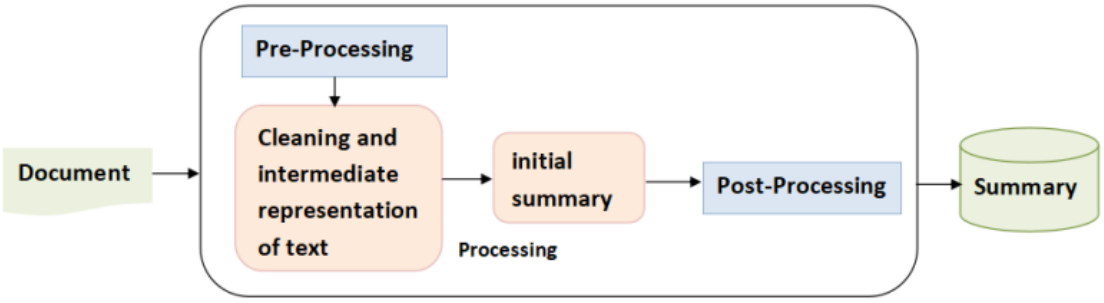

FIGURE 1.5: Abstractive Text Summarization Architecture.

(Bilingual Evaluation Understudy), and ROUGE (Recall-Oriented Understudy for Gisting Evaluation).

ROUGE provides information on recall and accuracy by calculating the word and phrase overlap between the generated summary and reference summaries. With n-gram precision, BLEU, which was first created for machine translation, assesses how similar the output text is to reference texts. By evaluating the semantic similarity between the generated and reference summaries using contextual embeddings from pre-trained language models such as BERT, BERTScore provides a more sophisticated understanding of the quality of the material.

Researchers and practitioners may assess the coherence, conciseness, and contextual correctness of abstractive summarization models in a systematic manner by combining various evaluation techniques. This guarantees that the summaries produced successfully convey the major ideas of the source material while preserving readability and linguistic quality. Through ongoing training and assessment, these models are able to provide summaries that are not only technically correct but also believable and interesting to readers.

Applications of Abstractive summarization includes news articles, papers, emails, chatbot transcripts, posts on social networks, and meeting minutes. This is called abstractive summarization [25]. It extracts important information from lengthy articles, making it easier to read and comprehend the major concepts without having to read the entire text [26], and facilitates the browsing and understanding of social networking sites such as Facebook and Twitter [27]. A thorough comprehension of the context in which the information is provided is necessary for

Abstractive summarization. Without this, it's easy to misunderstand subtleties or underlying meanings, which might result in summaries that are inaccurate or incomplete. There are ambiguous words or ideas in many writings that might be interpreted in several ways [28]. To effectively summarize such material, one must be able to determine the most pertinent interpretation given the context. When a document covers a variety of subjects or viewpoints, a good summary must logically incorporate all of this material. It might be difficult to maintain the author's intended message while balancing opposing points of view. It's crucial to maintain the author's original tone and approach when summarizing [29].

Trying to make technical information more approachable without compromising the original voice presents a unique challenge. It requires striking a delicate balance between being concise and providing sufficient detail to effectively convey the core ideas [28]. A well-crafted summary must be both informative and efficient, avoiding unnecessary verbosity while ensuring that critical information remains intact. Achieving this balance involves careful consideration of what aspects of the content are essential to preserve the intended message. Summarization is not merely a matter of condensing text—it often involves a transformation of knowledge. In this process, the meaning and structure of the original content must be faithfully retained, a task that becomes particularly difficult when dealing with specialized subjects that utilize complex terminology or discipline-specific jargon [29].

These fields often contain nuances that are crucial for accurate interpretation, and any misrepresentation or omission can lead to misunderstandings. Another significant challenge in summarization is managing cognitive load. When dealing with dense or highly technical material, readers may struggle to process information efficiently. A well-designed summary should mitigate this issue by distilling the most important points while presenting them in a way that is easier to grasp [30]. However, simplifying content without losing essential meaning is not always straightforward, as oversimplification can strip away necessary context or depth. Additionally, summarizing content that touches on cultural, historical, or domain-specific perspectives requires careful attention. There is a risk that crucial cultural

nuances may be lost or misrepresented, leading to distorted viewpoints or misunderstandings. This challenge is particularly relevant when summarizing materials that contain implicit cultural references or localized knowledge [31].

Finally, assessing the quality of a summary is inherently subjective. Determining what constitutes a "good" summary can be difficult, especially since different audiences have varying needs and expectations. A summary that is effective for an expert audience may not be as useful for a general reader, making it essential to tailor summaries to their intended audience while maintaining fidelity to the source material. Establishing clear evaluation criteria is therefore a critical step in ensuring that summaries fulfill their intended purpose effectively.

Abstractive summarization is an advanced natural language processing (NLP) task [31] that aims to generate a concise and coherent summary by interpreting and paraphrasing the core ideas and concepts within a document, rather than simply extracting key sentences or phrases [32]. This approach requires a deeper understanding of the text, allowing the model to rephrase or generate new sentences that preserve the original meaning in a more compact form. However, applying abstractive summarization to languages like Arabic and Urdu presents particular challenges due to several factors.

Both languages have highly complex and rich morphological structures, which means that words can take many forms depending on tense, number, gender, and case. Additionally, the diversity of dialects within Arabic and Urdu further complicates the task, as models must be able to handle a wide range of linguistic variations. Another significant challenge is the relative scarcity of digitalized linguistic resources in these languages when compared to English, which limits the availability of large-scale datasets, pre-trained models, and tools required for effective summarization. Despite these challenges, a general architecture for Arabic and Urdu text summarization systems has been proposed, as shown in Figure 1.6, which outlines the components and workflow necessary to handle these languages effectively [32].

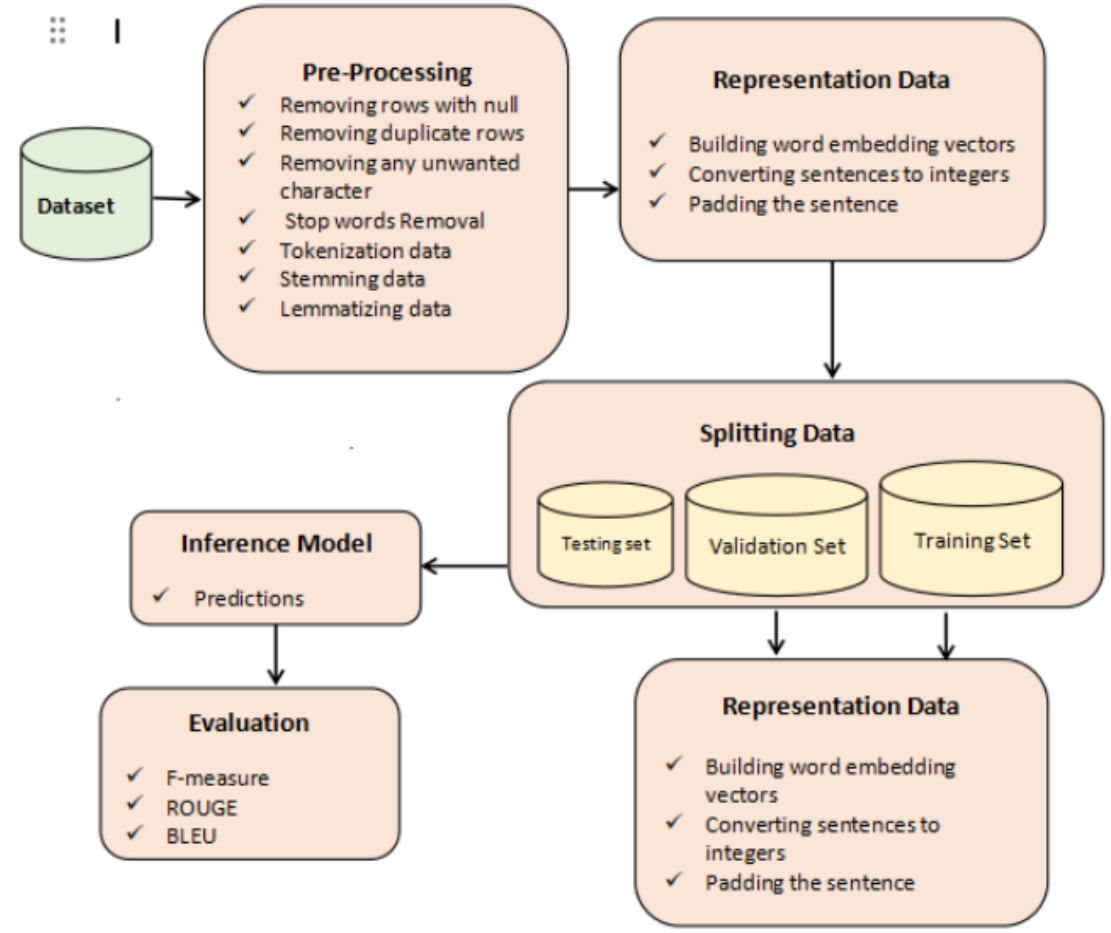

FIGURE 1.6: General Architecture of Arabic and Urdu Text Summarization System.

Figure 1.6 describes a whole pipeline for modeling, processing, and assessing a dataset, most likely for an NLP task. Raw data is imported for additional processing at the Dataset step. After that, the raw data passes through preprocessing, which entails a number of cleaning procedures, such as eliminating duplicates and rows with null values, filtering out undesirable characters, and deleting stop words to eliminate common but meaningless phrases. In order to make the data more consistent for analysis, it also goes through tokenization, which divides the text into discrete tokens, stemming, and lemmatization, which reduces words to their most basic forms [33]. Data representation stages are used to transform the preprocessed data into an appropriate format. In order to achieve consistency in model input, this entails constructing word embedding vectors that turn words into dense numerical representations, transforming sentences to integer sequences,

and padding these sentences to guarantee consistent length [32]. The fully represented data is divided into three distinct subsets: the training set, the validation set, and the testing set. The training set is used to train the model, providing the data from which the model learns patterns, relationships, and features relevant to the task at hand. During the training process, the validation set plays a crucial role in optimizing the model's performance. It is used to tune hyperparameters, adjust settings, and evaluate how well the model generalizes, helping to prevent overfitting and ensuring that the model's learned patterns are not specific only to the training data [33].

Finally, the testing set is used to provide an objective and realistic assessment of the model's performance. It consists of data that the model has not encountered during the training or validation phases, allowing for a true evaluation of how well the model performs on unseen data. This rigorous separation into training, validation, and testing subsets ensures that the model's performance is accurately measured and capable of generalizing to real-world scenarios.

To ensure consistent data formatting throughout, the representation procedure is applied to each data subset. Following training, the model moves on to the inference model stage, where it uses the test data to provide predictions [34]. The evaluation phase of the pipeline, which utilizes metrics like F-measure (to evaluate classification accuracy), ROUGE (to quantify overlap in tasks like summarization), and BLEU (to evaluate machine translation quality by comparing produced output to reference translations), comes at the end of the pipeline [35]. The model is trained on well-preprocessed data due to this organized pipeline, and its performance is reliably assessed using a variety of parameters [36].

Abstractive summarization in languages like Arabic and Urdu presents significant challenges, primarily due to their inherent morphological complexity. Both languages feature rich inflectional and derivational morphology, which increases the difficulty of accurately capturing meaning during the summarization process. Additionally, the scarcity of available resources—such as large, high-quality parallel corpora and pre-trained models—further complicates the development of effective summarization models.

Dialectal variations within these languages also add another layer of complexity, as models need to handle diverse linguistic forms and vocabulary. Orthographic complexities, including issues related to script variations and word segmentation, pose additional obstacles to accurate text processing. Compared to widely studied languages like English, where well-established evaluation metrics such as ROUGE, BLEU, and METEOR are frequently employed to assess text generation quality, Arabic and Urdu lack dedicated benchmarks that comprehensively capture their unique linguistic characteristics. The morphological richness, complex syntactic structures, and contextual dependencies of these languages make it particularly challenging to evaluate the quality of automatically generated summaries with existing metrics.

A significant challenge lies in the absence of standardized and language-specific evaluation frameworks tailored to Arabic and Urdu. Unlike English, where extensive research has led to the refinement of various assessment tools, these languages require specialized approaches that can account for aspects such as inflectional diversity, word order variations, and semantic nuances. The lack of such dedicated metrics not only hampers the development of high-quality summarization models but also makes it difficult to compare the effectiveness of different approaches in a consistent and reliable manner.

Developing robust evaluation criteria for Arabic and Urdu summarization remains an open research problem. Addressing this gap would require the creation of language-specific metrics that integrate advanced natural language processing techniques, including semantic similarity assessments, morphological analysis, and discourse coherence evaluation. Such advancements would enable more accurate and meaningful assessments of summarization performance in these linguistically rich and diverse languages.

Table 1.1 provides a summary of several text summarization studies, including information on their methodology, datasets, assessment measures, and advantages and disadvantages. In terms of text summary, the table lists a number of difficulties unique to Arabic and Urdu languages. The impact of each difficulty on the summarizing process is explained.

TABLE 1.1: Comparison of Summarizing Texts in Arabic and Urdu

Challenges	**Arabic**	**Urdu**
Complex Morphology	Complex morphological structure [32]. Root and inflection-based word building [32].	Difficult to provide accurate summaries [37]. Urdu's complex morphology [37].
Dialectal Variations	Arabic dialects that differ from Modern Standard Arabic (MSA) [32]. Difficult to create models [32].	Regional variances and dialects [38]. Due to these dialects, difficult to develop a model that is applicable to everyone [38].
Data Scarcity	Lacks extensive, high-quality annotated datasets [32]. Challenges for training and assessing abstractive summarization algorithms [32].	Large, high-quality annotated datasets are scarce [37]. Challenges for training and evaluating abstractive summarization models [37].
Ambiguity	Ambiguities that complicate summarization algorithms [32]. Difficult for algorithms to generate accurate and meaningful summaries [32].	Complicated by ambiguity in discourse [37]. Multiple interpretations of the same text [37].
Semantic Understanding	Complicated syntax and diverse dialects necessitate sophisticated NLP algorithms [37]. Difficult to achieve semantic understanding [37].	Complexity of accurately collecting and interpreting textual meaning [38]. Difficulty of achieving semantic comprehension [38].
Resource Restrictions	Shortage of Arabic-specific linguistic resources, tools, and pre-trained models [32]. Lack of resources [32].	Limited number of linguistic resources, tools, and pre-trained models [38]. Accurately capture the nuances of the Urdu language [38].
Named Entity Recognition (NER)	Exhibits spelling variances [32]. Complicates the process of identifying named entities [32].	Lack of standardization [37]. Uncertain word boundaries [37]. Lack of annotated data [37]. Interference from multiple languages [37].

Due to the complex morphologies of Arabic and Urdu, exact word creation and grammatical frameworks are necessary. Arabic's structure, which is built on roots and inflection, makes it difficult for summarization algorithms to correctly convey word meaning. Accurate summaries are also made more difficult by Urdu's grammatical patterns and word creation.

The distinct variances and geographical variations of Arabic and Urdu languages

provide difficulties for summarization models. A one-size-fits-all paradigm is challenging to develop since Urdu includes regional dialects and Arabic has several dialects. Arabic and Urdu suffer from a lack of huge, high-quality datasets that make it difficult to train efficient models, as well as a lack of comprehensive, high-quality annotated datasets that make it difficult to train and evaluate summarization methods.

The summary process can be hampered by ambiguity in Arabic and Urdu texts because algorithms may find it difficult to provide precise and insightful summaries because of the various ways that the same text can be interpreted, which might alter the intended meaning. Due to their intricate syntax and variety of dialects, Arabic and Urdu need sophisticated NLP algorithms for semantic interpretation. This is essential for efficient summarization and overcoming the difficulties presented by their different languages.

The efficacy of summarization systems is limited in Arabic and Urdu due to resource constraints. These include a lack of Arabic-specific linguistic resources, tools, and pre-trained models, as well as a lack of methods for precisely capturing linguistic subtleties. Arabic and Urdu have different Named Entity Recognition (NER) challenges: Arabic spelling variances make it more difficult to identify named things, while Urdu's lack of standardization makes it difficult to extract significant entities from text. Overall, these studies highlight the need for efficient methods that can handle a variety of material and linguistic complications, reflecting both notable developments and continuing difficulties in the field of text summarizing.

Chapter 2

Literature Review

2.1 Introduction

This chapter will offer a thorough and in-depth examination of the most recent and cutting-edge research, including theoretical, experimental, and empirical investigations. The main conclusions, approaches, and ramifications of recent research will be examined, and their importance and applicability will be critically assessed. Furthermore, the chapter will examine the contributions of prominent research papers on efficient administration and evaluate their influence on the state of the field. We will focus on how these basic and developing research support or contradict current beliefs and practices. To further emphasize their impact on the area, a detailed discussion of recent developments in administration, such as creative methods, developing frameworks, and real-world implementations, will be made. The objective of this chapter is to present a comprehensive grasp of the topic and its wider implications for future research and practice by combining these findings. The design and development of abstractive text summarization will be thoroughly examined, with a focus on languages like Urdu that have intricate scripts. A number of important areas that are essential to the development of summarization approaches will be covered in this study, including machine learning (ML), deep learning (DL), and extreme learning machines (ELMs). An examination of earlier strategies and technological advancements in the field will also be part of

the assessment, offering insights into their advantages and disadvantages as well as the development of strategies to deal with the particular difficulties presented by script languages that resemble Urdu.

A strong basis for comprehending current trends and pinpointing opportunities for additional development and innovation in abstractive summarization systems will be established by this thorough investigation. Abstractive text summarization will be achieved by reviewing several machine learning and deep learning technologies and methodologies, especially in Urdu-like script languages, such as text parsing and segmentation [39].

In [40], Awais & Nawab (2023) introduce the UATS-23 dataset and evaluate several techniques to address the challenges of developing models for the low-resource Urdu language. The study aims to enhance Urdu summarization tools without proposing a new model but highlights the effectiveness of the *GRU with Attention* model in improving the quality of Urdu text summarization. Attention-based GRU models perform well for abstractive summarization in Urdu, with the *GRU with Attention* model achieving the highest *ROUGE* scores (*ROUGE-1* = 46.7, *ROUGE-2* = 24.1, and *ROUGE-L* = 48.7). Conversely, transformer models like BART and GPT-3.5 faced issues with Urdu morphological compatibility and data scarcity. The paper addresses these challenges by providing a large Urdu dataset, establishing performance baselines, and offering insights into model selection and training for Urdu summarization.Future studies should focus on expanding dataset sizes, developing pre-trained models specifically for Urdu, exploring domain-specific and hybrid summarization strategies, and improving evaluation methods using semantic metrics and human assessments.

In [41], Raza & Shahzad (2024) research highlights the major obstacles to abstractive Urdu text summarization, including the lack of large-scale labeled datasets, linguistic complexity, and the complex syntactic structure of the Urdu language. To overcome these obstacles, the researchers use a labeled dataset of 19,615 documents and suggest a transformer-based framework for creating summaries, incorporating an encoder-decoder architecture for abstractive summarization and RoBERTa for contextual embeddings to improve the model's comprehension of

textual subtleties. The findings show that this method outperforms conventional extractive summarization methods in generating more accurate, coherent, and fluent summaries, with ROUGE-1 scores of 25.18, 12.5, and 23.27.

Despite these improvements, the study also addresses the ongoing drawbacks of existing Urdu abstractive summarization models, such as their comparatively poorer performance when compared to their English counterparts and their reliance on translated datasets, which can lead to inconsistencies.

To overcome these problems and increase the efficacy of Urdu text summarization, the study recommends the creation of high-quality, comprehensive datasets, the improvement of contextual embeddings, the use of diacritical markings to maintain meaning, and the introduction of datasets tailored to the length of the output.

In [42], Ali Raza & Hadia Sultan (2023) introduced an LSTM-based encoder-decoder model that was implemented in Python and TensorFlow for abstractive summarization of Urdu text. ROUGE-1, ROUGE-2, and ROUGE-L all had average F1 scores of 0.43, 0.25, and 0.23, respectively. However, the study has drawbacks, including its dependence on a particular news collection, the subjectivity of summaries created by humans, and difficulties in handling the subtleties of Urdu grammar. Moreover, it is not scalable and overfits, suggesting that further study is required. Future research ought to expand datasets, integrate a range of metrics, improve preprocessing techniques, look into complex models, address language-specific problems, take user input into account, become multilingual, and employ knowledge graphs.

In [43], Asif Raza (2024) presents a hybrid method for generating abstractive summaries of Urdu text, combining extractive summarization techniques with the BERT model. The study uses a small dataset of 50 articles from sources like BBC Urdu and Jang News, covering topics like health, news, sports, and technology. The study achieved varying compression rates, with extractive summaries at 38–41%, hybrid summaries at 33%, and abstractive summaries at 25%. Expert evaluation emphasized the importance of human assessment over automated metrics, resulting in fluent and meaningful summaries. However, the study has several

limitations, including a small dataset, variable summary quality, pre-training constraints, computational resource challenges, subjectivity, and the neglect of dialect variations and scalability. Future research should focus on expanding datasets, developing multidocument summarization techniques, incorporating user feedback, and enhancing performance and accessibility by leveraging transfer learning and integrating linguistic features.

In [44], M. Asif & S. A. Raza (2022) address the inherent complexity of the Urdu language, a research team created a sophisticated natural language processing (NLP) model that makes use of bidirectional encoding and a semantic extractor. To improve fluency, coherence, and semantic correctness, the model was especially created in light of Urdu's complex morphology, flexible word order, and contextual dependencies. With an emphasis on tackling language issues including grammatical errors, missing diacritical marks, and contextual ambiguity, the method was evaluated on a carefully selected dataset of Urdu news articles and blogs. The bidirectional encoder-decoder system at the heart of this technology efficiently extracts contextual information from both previous and subsequent text segments, allowing for a more complex and semantically rich comprehension of the language. The summarization method is further improved by adding a semantic extractor, which finds important terms and guarantees that important information is retained.

The bidirectional technique significantly outperformed conventional rule-based and statistical summarizing approaches, according to performance evaluations utilizing popular NLP metrics including recall, accuracy, F1 scores, and ROUGE scores. These findings highlight how sophisticated methods such as bidirectional encoding might enhance the caliber of summaries that are produced. The report does, however, also recognize a number of shortcomings that need to be fixed in order to go forward. The robustness of the preprocessing stages is a significant difficulty since precise text normalization, tokenization, and diacritical restoration are critical to the overall efficacy of the model.

The performance of the model may be adversely affected by any flaws in these preprocessing steps, especially in languages like Urdu that have intricate script systems. The study's very short dataset size is another significant drawback that

limits the model's capacity to generalize well to other Urdu text domains. The model's resilience and adaptability to various situations may be improved by enlarging the dataset to incorporate a greater range of text sources. Furthermore, the model's performance has not yet been evaluated in a wider, more varied language context due to the absence of multilingual and cross-lingual evaluations, which restricts the model's use in international contexts.

Multilingual characteristics would also enable more efficient usage in multilingual voice recognition systems and assist the model manage code-mixed text, which is when different languages are utilized in the same sentence or document. An further disadvantage that has been noted is the computational burden of the bidirectional encoding process, which may make it difficult to scale the model to big datasets or real-time applications. Practical implementation will require optimizing the model to increase its effectiveness and lower its computing burden. A more thorough human review procedure that would evaluate the summaries' qualitative qualities, such coherence, readability, and relevance, is also required, the research stresses. The report offers many ideas for future research areas to get over these restrictions. Key areas for development include adding more varied text sources to the dataset, improving pretreatment procedures to increase the resilience of preprocessing, and implementing reinforcement learning strategies to improve summarization quality repeatedly. Additionally, adding multilingual and cross-lingual characteristics might greatly boost the model's versatility and allow it to function well in a larger variety of languages, including ones with speech-based input and mixed linguistic content. In [45], Nida Shafiq & Isma Hamid (2023) propose a method for abstract text summarization in low-resource languages like Urdu using self-attentive transformer-based models. They use a custom Urdu news article summary dataset of 76,500 article summary pairs to demonstrate comparable performance to high-resource languages. The urT5 model, developed using multilingual transformer models, outperforms high-resource language models like PEGASUS and BART on the dataset. Despite its small size, the model outperforms high-resource language models like PEGASUS and BART. However, the urT5 model has limitations, including a small dataset, potential biases, and limited applicability across low-resource languages. Future research should expand

the Urdu dataset, improve model scalability, apply the urT5 model to low-resource languages, and address biases in pre-trained models for improved performance.

In [46], Ali Faheem & Faizad Ullah (2024) discuss the challenges in multimodal summarization for low-resource languages like Urdu, recommending the development of diverse, high-quality datasets like UrduMASD. The UrduMASD dataset uses pre-trained multilingual models to improve performance in low-resource languages like Urdu, improving *ROUGE* scores by 2.6%. The paper explores multimodal abstractive summarization using the UrduMASD dataset, comparing text-only models, text and visual data models, and multimodal fusion models.

The study demonstrates that incorporating visual data into multimodal summarization improves *ROUGE* scores by 2.6%, outperforming existing datasets in terms of abstractivity, compression, and coherence. However, the study identifies several limitations, including limited diversity in the dataset, transcription errors, challenges related to code-switching, and potential pretraining requirements. Future research should focus on expanding datasets, enhancing automatic speech recognition systems, optimizing models, and exploring real-world applications.

2.2 Comparative Analysis and Survey of Existing Techniques

A comprehensive comparison will be conducted between the proposed summary technique and existing Urdu abstractive text summarization systems documented in the literature. This analysis aims to highlight the unique strengths, potential limitations, and distinguishing features of the proposed approach in contrast to current methodologies.

The comparative study will focus on multiple critical dimensions, including the underlying summarization techniques, the diversity and size of training and evaluation datasets, and key linguistic factors that influence summary quality. Specifically, it will examine challenges unique to the Urdu language, such as its complex

TABLE 2.1: Comparison between different Techniques for Abstractive Urdu Summarization (Part 1)

Authors & Year	**Methodology & Technique**	**Dataset**	**Evaluation Metrics**	**Merits & Demerits**
Kuan-Yu Chen, 2024 [47]	RNN Language model	CNN/Daily Mail, DUC, TAC (news articles)	ROUGE metric	Not statistically significant. Performs better than existing methods.
Ramesh Nallapati, 2019 [48]	Attention mechanisms, Sequence-to-sequence	CNN/Daily Mail, Gigaword, DUC (various domains)	ROUGE metric	Produces consistent results. Offers comprehensive techniques.
Peter J. Liu, 2022 [[49]]	Pointer generator, Conventional attention-based seq2seq	CNN/Daily Mail (news articles)	ROUGE-1, ROUGE-2, ROUGE-L, BLEU	Repetition was prevented. Restrictions were imposed on coining new phrases.
Sebastian Gehrmann, 2023 [50]	Content selector	CNN/Daily Mail, Gigaword, XSum, PubMed (research papers)	ROUGE, F-measure	Enhances text compression. Selects essential phrases for extractive summary. Optimizes content selection for abstractive summarization.
Rishabh Katna, 2023 [51]	Latent Semantic Analysis (LSA), Feature-based approach	DUC, CNN/-Daily Mail, Gigaword, PubMed	Precision, Recall, F-measure	Uses feature-based approaches. Targets specific audience.
Chenguang Zhu, 2023 [52]	Hierarchical Network, Cross-Domain Pretraining	AMI Corpus, ICSI Meeting Recorder, MMD Corpus, DyGIE++ Corpus (meeting transcripts)	ROUGE metric	Rewritten text shows better results than earlier methods. Document contains multiple transcripts, one for each speaker.
Jinge Yao, 2023 [53]	Joint summary generation	Multi-News, TAC Multilingual Summarization, XSum, Wiki, Multilingual-LS	ROUGE metric	Complicated morphology of the Urdu language. Scarce resources available for the language.
Ammar Mohammed, 2023 [54]	Statistical-Based methods, Using machine learning	TAC Arabic Summarization Corpus	Precision, Recall, F-measure	Complex task due to the language's complexity and unique linguistic components. Understanding Arabic text summarizing approaches is crucial for effective research.
Aziz Qaroush, 2023 [55]	K-Medoid clustering	AMDS (domain-specific corpus)	ROUGE metric	Improves productivity and effectiveness in text summarization. Evaluation metrics play a crucial role.

TABLE 2.2: Comparison between different Techniques for Abstractive Urdu Summarization (Part 2)

Authors & Year	**Methodology & Technique**	**Dataset**	**Evaluation Metrics**	**Merits & Demerits**
Asmaa Elsaid, 2024 [56]	Transformer encoder-decoder network	DUC, CNN/-Daily Mail, Gigaword, PubMed	ROUGE-1, ROUGE-2, ROUGE-L, BLEU	Using modern deep learning algorithms demands significant memory processing capacity. Large-scale or resource-constrained projects are challenging to complete.
Hassan Raza, 2023 [57]	Neural network architecture, Attention mechanisms	DUC, CNN/-Daily Mail	ROUGE metric	Lack of information on the dataset. Needs better evaluation measures for summaries.
Muhammad Awais, 2023 [58]	Transformer-based encoder-decoder model	DUC, CNN/-Daily Mail	ROUGE-1, ROUGE-2, ROUGE-L, BLEU	Model's effectiveness depends on the dataset's size and variety. Encoder-decoder method is suitable for Abstractive summarization.
Raghav Jain, 2023 [59]	Reward function, Word Mover Distance (WMD), WMD-based reward function	SEPS, SEPS-AS	ROUGE metric	Method outperforms existing approaches in biomedical text summarizing. Limited applicability to other domains or languages.
Dawood Ashraf Khan, 2022 [60]	Bio-semantic models	PubMed, PMC-OAS	ROUGE metric	Improves biomedical text summarization by maintaining text semantics. Not transferable to other domains outside the biomedical sector.

syntactic structures, rich morphological variations, and the presence of multiple dialects, all of which can significantly impact the performance of abstractive summarization models.

In order to assess the effectiveness of the proposed approach, the study will employ standard performance evaluation metrics, including ROUGE scores, BLEU scores, and other relevant assessment measures. These metrics will provide a quantitative

basis for determining the quality of the generated summaries relative to state-of-the-art summarization techniques.

By systematically comparing the performance of the proposed method with cutting-edge approaches, the study will offer valuable insights into areas where the system excels as well as aspects that may require further refinement. This rigorous evaluation will contribute to a deeper understanding of the advantages and shortcomings of the proposed methodology, facilitating future enhancements that could optimize its applicability for real-world Urdu text summarization tasks.

Table 2.1 provides a summary of several text summarization studies, including information on their methodology, datasets, assessment measures, and advantages and disadvantages. Among the methodologies are neural networks (RNNs, transformers), attention mechanisms, and statistical techniques.

The datasets range from Gigaword and CNN/Daily Mail to specialized collections like the Arabic Summarization Corpus and PubMed. In addition to precision, recall, and F-measure, the majority of research use ROUGE metrics. While the drawbacks include high computational requirements, a lack of dataset information, and difficulties applying the model across various languages and domains, the advantages are enhanced performance over current approaches, consistency, and handling of complex languages or specialized domains. Overall, these studies highlight the need for efficient methods that can handle a variety of material and linguistic complications, reflecting both notable developments and continuing difficulties in the field of text summarizing.

2.3 Identified Research Gaps

Although the Abstractive Urdu text summary approach rephrases content logically to extract important information from the original source [61], it is vital to address ignored semantic concerns in the base paper [12]. The suggested model has yielded better and more advantageous outcomes by resolving these problems.

The study's main goal is to use an Abstractive text summarizing model with a semantic extractor to enhance the readability and general comprehension of summaries written for the Urdu language. In addition to improving readability, this will extract the semantic meaning of the text.

2.4 Problem Statement

Previous studies on Urdu abstractive text summarization have made significant progress towards producing concise and coherent summaries [47]. However, there is a significant gap in capturing and maintaining the semantics of the original text [51]. The underlying semantics of Urdu text were not successfully extracted and reflected by previously suggested methodologies. It produces inconsistent, ambiguous statements that lack sufficient accuracy as well as clarity.

2.5 Research Questions

By implementing the proposed system, the following queries will be answered:

1. Can semantic extraction improve performance of abstractive urdu text summarization?
2. How can the design of the encoder-decoder architecture be optimized to improve the performance of abstractive text summarization in Urdu?
3. How does the integration of semantic extractors within the model pipeline enhance the quality of the generated summaries?

2.6 Objectives of the Research

The first suggestion for abstracting English text summarization using deep learning techniques was made in 2015 [62]. The main goal of this study is:

- Explore the algorithms (e.g. neural networks, attention mechanisms, etc.) that are used by the model to capture semantic information and their effectiveness in producing meaningful summaries.

- Optimize the encoder-decoder architecture for Abstractive text summarization in Urdu by implementing attention mechanisms or pretrained language models.

- Evaluate the overall impact of integrating semantic extractors on the resulting summaries in terms of coherence, and relevance.

Chapter 3

Research Methodology

3.1 Introduction

In NLP, automated text summarizing has become more popular as a way to distill long texts into brief summaries [1]. By combining and summarizing information, abstractive processes provide flexibility in contrast to traditional methods, which are mostly extractive [52]. However, these methods are not often applied to languages like Urdu that have complicated morphology and semantics [62]. This paper presents a novel approach to Abstractive text summarization in Urdu using several encoders, a semantic extractor in deep learning, and a single decoder.

3.2 Proposed Research Methodology

A multi-step research strategy was used in order to thoroughly explore the study subject and offer answers to the research questions listed in Section 2.5. In order to guarantee comprehensive investigation and analysis, this method was created to methodically address every facet of the study. Figure 3.1 shows the research approach's systematic growth, highlighting the connections between each step, which begins with the literature review and progresses through data collection, model creation, experimentation, and validation. Every stage builds on the one

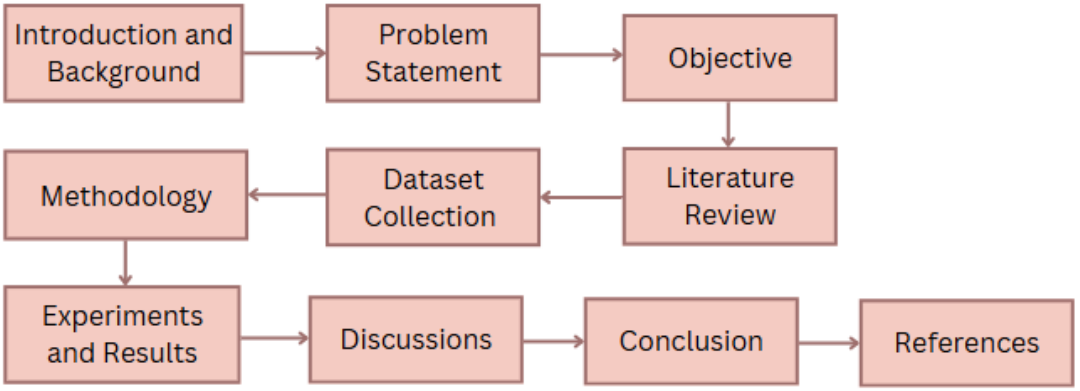

FIGURE 3.1: Research Methodology

before it, guaranteeing a coherent and rational flow that directly tackles the study topic and the particular issues brought up in the research questions. The key actions and techniques required in carrying out the research are highlighted in the following descriptions of the several stages that make up the architecture of the research methodology.

Figure 3.1 illustrates the architecture of research methodology. The Introduction and Background section provides a brief overview of the topic's history, discusses the idea of text summarization, and distinguishes between expressive and abstractive forms. They also outline the investigation's scope and goal. The section on "Problem Statement" highlights the primary issues and challenges with the text outlining strategies already in use, emphasizing their flaws and gaps in performance, accuracy, and efficiency, and illustrating the necessity of finding solutions. Objectives With a focus on enhancing text summarizing models, boosting the coherence and accuracy of summarized texts, contrasting with existing methods, and validating with pertinent metrics, the objectives are clearly articulated.The section on literature reviews earlier research and text summarizing techniques.

To set the stage for the development of new models, it highlights the benefits and drawbacks of various algorithms and approaches, reviews significant papers, and discusses them. In order to ensure accurate model evaluation, preprocessing methods, source identification, dataset splitting, and other steps involved in gathering and getting ready data for training and testing the summarization models are described. The methodologies and approaches for developing and evaluating

text summarizing models are described, providing a comprehensive basis for the study. These include model selection, training, assessment metrics, and baseline comparison. The findings of many models and analyses are presented in the results and experiments section, along with experimental configurations and settings. It uses statistical tools to compare outcomes with baseline models in order to verify significance. Discussions of the results and consequences focus on their relevance, comparisons with other research, advantages and disadvantages of the suggested models, and recommendations for improvement.

In conclusion Outlining key findings, highlighting contributions to the field, and offering ideas for future research areas, an overview of the study's findings and contributions is given.A comprehensive list of all the sources utilized in the research is put together using a standard citation format. Books, journals, research papers, and other relevant resources are included in this list.

The proposed architecture is composed of multiple components.The entire summary process is built around the input text, which consists of Urdu papers or articles. A critical stage in text analysis is preprocessing, which includes managing punctuation, normalization, and tokenization. Normalization standardizes format, tokenization divides text into manageable chunks, and punctuation management adds readability and clarity to sentences. This procedure makes sure that the material is legible and easily understood. Feature extraction, which may include sentence scoring and key phrase extraction, is the process of removing important words or phrases from a text. Assigning weights to sentences according to their significance and identifying the substance of the text depend heavily on these procedures.

The most important lines from the feature extraction stage are chosen as a first step in the extractive summary creation process, which is before the final abstractive summary is created. The suggested approach employs a multi-layer encoder-decoder paradigm to generate abstractive summaries. While the decoder creates the summary, the encoder interprets the incoming text. While the decoder paraphrases the original information, the encoder records the semantic meaning. To

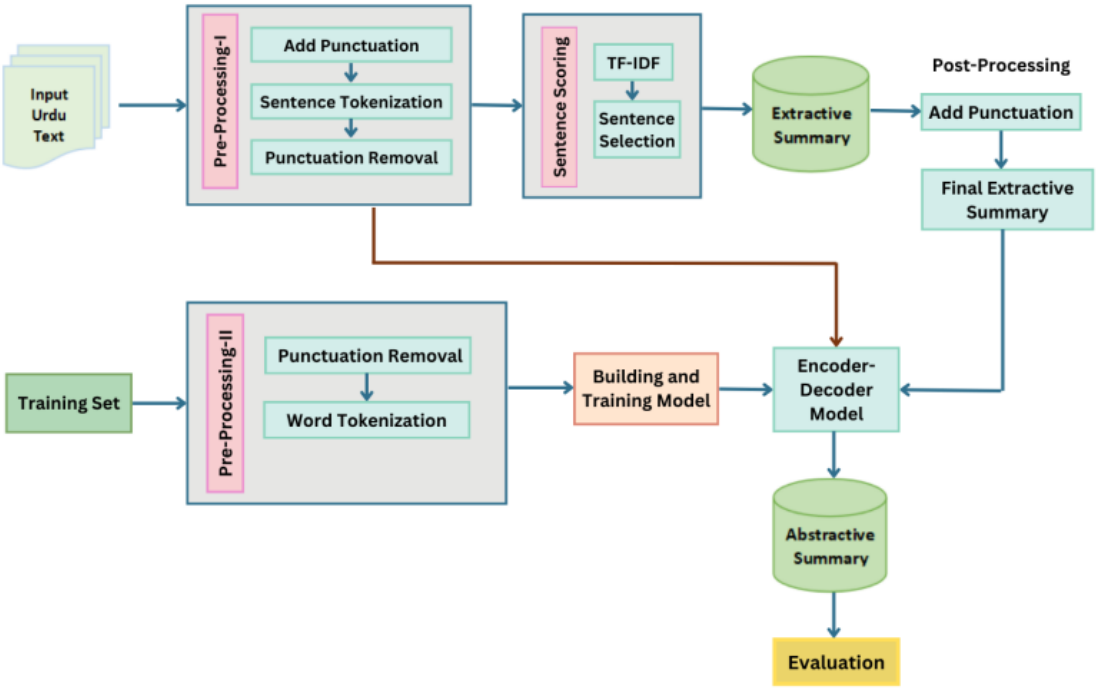

FIGURE 3.2: Proposed Architecture for Abstractive Urdu text summarization methodology.

guarantee coherence and relevance in the output, an attention mechanism is used to concentrate on pertinent portions of the input text.

In order to effectively paraphrase the information while maintaining its meaning, the framework creates an abstractive summary, which is a condensed version of the original text that may contain new words and sentences. In order to compare the quality and relevance of the summary to reference summaries, the model's performance is evaluated using metrics such as ROUGE, BLEU, and BERTScore.The framework could have a feedback loop where assessment findings inform model improvements, encouraging an iterative process of ongoing summarization quality improvement.The suggested methodology's framework is shown in Figure 3.2.

3.2.1 Pre-Processing

Any NLP task usually starts with text preprocessing [63]. Many open-source programs are available for the English language [64]. When it comes to Pashto, Persian, Arabic, and other closely related languages like Urdu, it is still exceedingly difficult [65]. For the Urdu language, there are a number of e-libraries available,

but preprocessing still needs a lot of effort because of their poor accuracy [66]. Punctuation marks clarify sentence meanings, enhancing readability and understanding [67]. In Urdu text, absence of punctuation can cause ambiguity. Proper punctuation improves readability for humans and machines [68] , as shown in Figure 4.3. The function of punctuation in Urdu phrases and its importance in relation to text summarization and Natural Language Processing (NLP) are the subjects of Figure 4.3. In Urdu texts, punctuation is essential for readability, clarity, and meaning. It aids in defining the structure of sentences by highlighting pauses and connections. Sentences that are punctuated correctly may be interpreted differently. Readability is also improved by proper punctuation, which enables readers to assimilate information more quickly. Because readability influences how well NLP models perform on tasks like sentiment analysis, translation, and summarization, it is very important. For feature extraction, sentence boundary recognition, and text structure understanding, punctuation is crucial in machine learning and natural language processing (NLP). Tokenization, text segmentation into words or phrases, and the creation of logical summaries all depend on proper punctuation. Significant preprocessing difficulties exist, particularly for tasks like tokenization and in languages like Urdu. In languages with extensive morphology and syntax, such as Urdu, punctuation greatly minimizes ambiguity in unpunctuated writing. The way a comma is placed can drastically alter how a statement is understood. Particularly for low-resource languages like Urdu, punctuation improves readability, clarity, and the efficiency of machine learning models in processing and comprehending text.

Tokenization is one of the most important phases in the prior treatment [69]. Tokenization is the process of dividing a text, whether it be a sentence, phrase, paragraph, or the entire document, into distinct or unique terms [70], as shown in Figure 3.4. A critical stage in text summarization and NLP activities is the tokenization of Urdu sentences, as shown in Figure 4.4. In the field of Urdu NLP, tokenization is an essential procedure. It entails breaking a text up into separate parts, such words and phrases, in order to determine its start and finish. Because of Urdu's intricate script and morphology, which may make it challenging to recognize word boundaries without obvious marks, this is particularly crucial. In addition

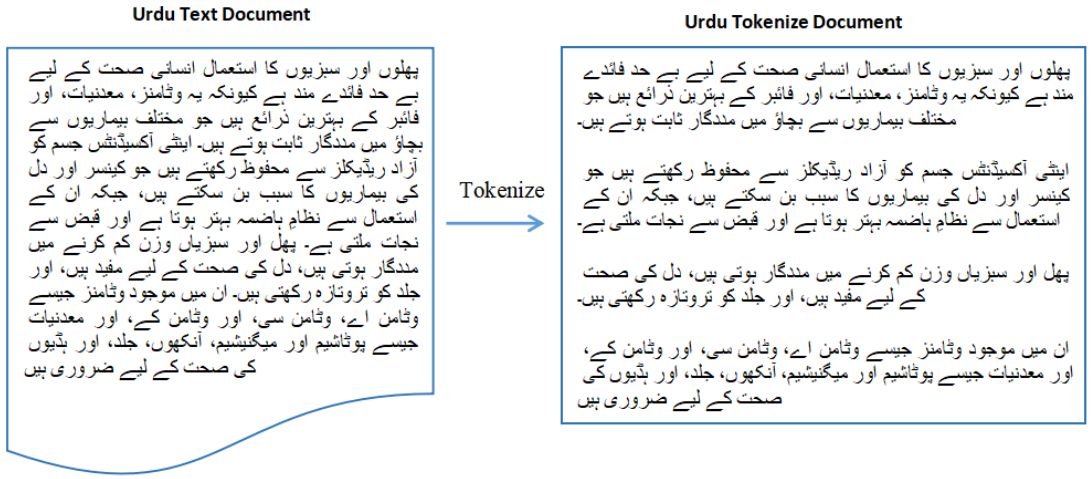

FIGURE 3.3: Punctuation of Urdu Sentence.

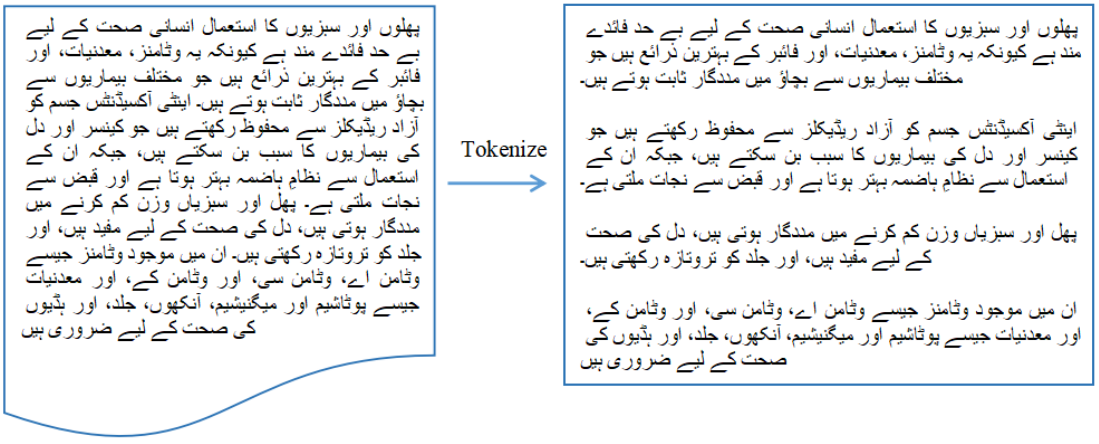

FIGURE 3.4: Tokenization of Urdu Sentence.

to making text processing easier, tokenization improves the effectiveness of NLP models' text analysis. Through the division of the text into uniform pieces, it aids in standardizing spelling, dialects, and informal use variances in Urdu.

Tokenization plays a crucial role in the interpretation and understanding of text by preserving the word order, which is vital for grasping its underlying structure and semantics [71]. Proper tokenization is an essential preprocessing step in natural language processing (NLP), as it divides raw text into smaller, meaningful units that can be more easily analyzed and processed by computational models. The tokenization process generally unfolds in two main stages.

In the first stage, lengthy and complex utterances are broken down into smaller, more manageable units, such as individual sentences and words [72]. This step

is crucial for ensuring that the text is segmented in a way that aligns with the natural linguistic structure, making it easier for subsequent stages of processing to operate effectively.

The second stage of tokenization focuses on breaking down phrases into discrete elements by identifying specific linguistic markers, such as sentence and word boundaries, punctuation marks, and other delimiters [73]. These markers serve to clearly define the boundaries between words and phrases, ensuring that the text is segmented in a way that retains its intended meaning, structure, and coherence. This segmentation is fundamental for downstream tasks like parsing, part-of-speech tagging, and machine translation, where understanding the relationship between individual tokens is critical for accurate analysis and interpretation.

The beginning and ending points of words are determined based on these word boundaries, which are established through the tokenization process [93]. Additionally, punctuation marks, such as question marks (?), exclamation points (!), and full stops (./-), serve as paragraph separators, further aiding in segmenting the text into logical and meaningful units [74]. These tokenization steps are essential for preprocessing text, enabling accurate parsing and analysis for further computational tasks.

These tokenized sentences serve as the foundational units for subsequent text processing steps, one of the most important of which is punctuation removal. Punctuation marks, while important for human comprehension, often introduce unnecessary complexity for computational models, which can distract from the core content of the text. By eliminating punctuation, the focus shifts entirely to the primary linguistic elements, allowing language models to more effectively capture word semantics and underlying linguistic patterns [75]. This normalization step ensures that the text is represented in a more uniform way, reducing the variability caused by different punctuation marks and allowing the model to prioritize the meaningful syntactic and semantic relationships between words and phrases.

Furthermore, punctuation removal contributes to improving the consistency of textual representation, creating a cleaner input for models to process. This step

is especially important in tasks such as machine translation, text generation, and sentiment analysis, where clarity in word relationships is crucial for accurate predictions. Additionally, removing punctuation can also enhance the overall readability of the text. By streamlining the content, it allows both human readers and computational models to focus on the most relevant information, reducing distractions caused by non-essential marks and improving the ease with which the text is interpreted [76].

3.2.2 Extractive Summary

Once the pre-initialization procedure is completed, the relevant properties of the text are extracted. Extractive summarization is the technique of selecting key words, sentences, or phrases from the original text based on a predefined criterion or standard. This method involves identifying the most informative portions of the text that accurately capture its essence. The result is a concise summary that preserves the original meaning, while omitting less significant details, ensuring that the core ideas are effectively conveyed [77]. By focusing on crucial elements, extractive summarization aids in creating summaries that are both comprehensive and faithful to the original content.

3.2.2.1 Sentence Weight Algorithm

After preprocessing, the text is prepared for the next critical step in the summarization process: feature extraction. In extractive summarization, relevant phrases and sentences are selected based on a range of feature sets designed to capture the essential ideas and meaning of the original content [78]. This stage involves carefully identifying key sentences and phrases that best represent the core message of the document, ensuring that the generated summary reflects the most important information without introducing redundancy.

Each source phrase is assigned a weight, which indicates its significance within the context of the document. The weight is determined by various factors such as

keyword frequency, sentence position, syntactic structure, and semantic relevance. Sentences containing phrases with higher weights are ranked more highly in the summary generation process [79]. This weighted ranking ensures that the most important and informative parts of the text are prioritized in the final extractive summary, helping to produce a concise yet comprehensive representation of the original content.

The weight assignment is typically determined using sentence weight algorithms, which are statistical methods designed to evaluate the relevance of individual phrases and sentences in relation to the overall document. These algorithms assess the significance of each sentence based on various factors, such as frequency of important keywords, sentence position, and syntactic structure. One common approach to sentence weighting is to measure the proportion of substantive (meaningful) words to the total number of words in the sentence. Sentences that contain a higher percentage of substantive words—such as nouns, verbs, and adjectives that carry more meaning—are given higher weights, indicating their importance for inclusion in the final summary [80]. This method ensures that the extractive summary remains concise while retaining the most relevant and informative content.

According to Eq. (1), let W = L1, L2, L3,... LN represent the given Urdu source text, where n stands for the total number of sentences and Li for a single phrase [81]. A token is created for each word, as seen in figure 4.5. The weight w of each phrase Li is computed using the following formula [82]:

$$W_i = \frac{\text{Number of Filtered Words in } L_i}{\text{Total Number of Words in } L_i} \tag{3.1}$$

3.2.2.2 Word Frequency Algorithm

An other popular statistical method for assessing a term's significance inside a text is Term Frequency-Inverse text Frequency (TF-IDF), which offers important information about which phrases are most pertinent. The method determines each

Sentence	Score
پھلوں اور سبزیوں کا استعمال انسانی صحت کے لیے بے حد فائدے مند ہے کیونکہ یہ وٹامنز معدنیات اور فائبر کے بہترین ذرائع ہیں جو مختلف بیماریوں سے بچاؤ میں مددگار ثابت ہوتے ہیں	5.5100
اینٹی آکسیڈنٹس جسم کو آزاد ریڈیکلز سے محفوظ رکھتے ہیں جو کینسر اور دل کی بیماریوں کا سبب بن سکتے ہیں جبکہ ان کے استعمال سے نظام ہاضمہ بہتر ہوتا ہے اور قبض سے نجات ملتی ہے	5.4148
پھل اور سبزیاں وزن کم کرنے میں مددگار ہوتی ہیں دل کی صحت کے لیے مفید ہیں اور جلد کو تروتازہ رکھتی ہیں	4.3354
ان میں موجود وٹامنز جیسے وٹامن اے، وٹامن سی، اور وٹامن کے، اور معدنیات جیسے پوٹاشیم اور میگنیشیم، آنکھوں، جلد، اور ہڈیوں کی صحت کے لیے ضروری ہیں	4.0266

FIGURE 3.5: Sentence Scoring.

term's weight by taking into account two factors: the term's frequency inside a particular document (Term Frequency, or TF) and its inverse frequency, or Inverse Document Frequency, or IDF, across all documents in the corpus [83]. More attention is given to terms that appear often in one document but infrequently in others since a term's weight rises with its frequency in the document and falls if it appears frequently in multiple papers.

This guarantees that terms that are more distinctive and document-specific are given greater weight. The study of token occurrences in the text [84] is visualized in Figure 3.6, which helps to highlight the most unique phrases by differentiating between those that are less prevalent in the larger corpus and those that occur often inside the document.

Suppose w to be word1, word2,... wordn. The total number of words in the document is Wn, and according to Eq. (2), this is the total number of words.

$$\mathrm{TF} = \frac{W}{W_n} \tag{3.2}$$

Compute the total number of credentials Dn by the document frequency Df to get the IDF [85], as shown by Equations (3) and (4).

$$\mathrm{TF} = \log\left(\frac{D_n}{D_f}\right) \tag{3.3}$$

Sentence	Score
پھلوں اور سبزیوں کا استعمال انسانی صحت کے لیے بے حد فائدے مند ہے کیونکہ یہ وٹامنز معدنیات اور فائبر کے بہترین ذرائع ہیں جو مختلف بیماریوں سے بچاؤ میں مددگار ثابت ہوتے ہیں	5.5100
اینٹی آکسیڈنٹس جسم کو آزاد ریڈیکلز سے محفوظ رکھتے ہیں جو کینسر اور دل کی بیماریوں کا سبب بن سکتے ہیں جبکہ ان کے استعمال سے نظام ہاضمہ بہتر ہوتا ہے اور قبض سے نجات ملتی ہے	5.4148
پھل اور سبزیاں وزن کم کرنے میں مددگار ہوتی ہیں دل کی صحت کے لیے مفید ہیں اور جلد کو تروتازہ رکھتی ہیں	4.3354

FIGURE 3.6: Selected Sentences.

$$\text{TF-IDF} = \text{TF} \times \text{IDF} \tag{3.4}$$

The Term Frequency-Inverse text Frequency (TF-IDF) value of each token may be used to analyze the relevance of a word inside a text. By taking into account how frequently a word occurs inside a document in comparison to how often it occurs throughout the whole dataset, this statistical measure aids in identifying the most significant terms in a text. Sentences are arranged according to their values in ascending order following the computation of the TF-IDF scores. Choosing phrases with high TF-IDF scores is essential for producing an extensive and educational summary [86], as shown in Figure 3.7, since higher TF-IDF values signal more relevance.

The most pertinent and distinctive information from the original text is preserved in the extracted material thanks to this method. In order for the model to produce an abstractive summary, it must first go through a training phase in which it learns to produce meaningful and cogent summaries instead of just extracting important sentences. Figure 3.8 shows the full training and assessment procedure, showing how the transformer-based model is trained on a structured dataset and then evaluated using test data that has not been seen to ascertain its generalization capabilities.

Figure 3.8 illustrates the workflow of a summarization model. A collection of documents and their accompanying summaries are displayed in the "DATASET" part of the Dataset Preparation section. The model will use this dataset to identify patterns for summarization since it offers the source texts together with the desired

پھلوں اور سبزیوں کا استعمال انسانی صحت کے لیے بے حد فائدے مند ہے۔
کیونکہ یہ وٹامنز معدنیات اور فائبر کے بہترین ذرائع ہیں۔ جو مختلف بیماریوں
سے بچاؤ میں مددگار ثابت ہوتے ہیں۔ اینٹی آکسیڈنٹس جسم کو آزاد ریڈیکلز سے
محفوظ رکھتے ہیں۔ جو کینسر اور دل کی بیماریوں کا سبب بن سکتے ہیں۔
جبکہ ان کے استعمال سے نظامِ ہاضمہ بہتر ہوتا ہے۔ اور قبض سے نجات ملتی
ہے۔ پھل اور سبزیاں وزن کم کرنے میں مددگار ہوتی ہیں۔ دل کی صحت کے
لیے مفید ہیں۔ اور جلد کو تروتازہ رکھتی ہیں۔ ان میں موجود وٹامنز جیسے
وٹامن اے وٹامن سی اور وٹامن کے اور معدنیات جیسے پوٹاشیم اور میگنیشیم
آنکھوں جلد اور ہڈیوں کی صحت کے لیے ضروری ہیں۔

FIGURE 3.7: Extractive Summary

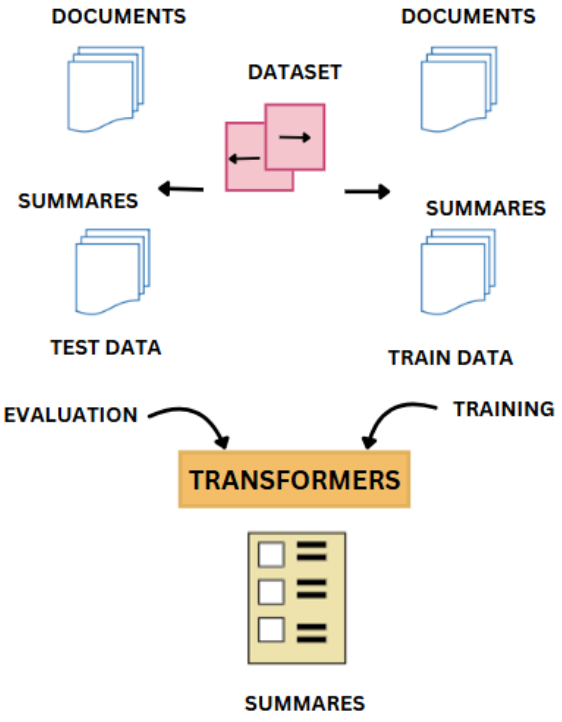

FIGURE 3.8: Process of Training and Evaluation.

summaries. Train Data and Test Data are the two sections of the dataset. The model is trained using training data. The transformer model will learn from these texts and their summaries. The purpose of test data is to assess the model's performance following training. The model's ability to produce correct summaries from unseen documents is tested using the test data's summaries.In the training phase, the arrow from "Train Data" to "Transformers" is used. In this phase, the transformer model examines patterns in the training data to learn how to map

documents to summaries. This entails fine-tuning the internal parameters of the model to provide summaries that closely match those seen in the training dataset. In the evaluation phase, the arrow from "Test Data" to "Transformers" illustrates how the model's performance is assessed using unseen test data after it has been trained. Once the model has undergone the training process, it is tested on new, previously unseen data to determine its ability to generalize and generate accurate, relevant summaries. The purpose is for the model to perform well on unfamiliar texts, demonstrating its robustness and adaptability to diverse input data. The output produced from the "Transformers" block is referred to as "SUMMARIES," which represents the model's final generated text after processing the input data.

These "SUMMARIES" are evaluated to determine how effectively the transformer-based model has learned from the training data. The effectiveness of the model is assessed by examining the quality, coherence, and relevance of the generated summaries, focusing on whether the summaries faithfully capture the main points and essential information of the original text.

In this context, the "TRANSFORMERS" block refers to a transformer-based model, such as BERT, GPT, T5, or similar architectures, which are central components in the summarization process. Transformers are a class of powerful neural network architectures that have revolutionized various natural language processing (NLP) tasks, including text summarization. Their ability to capture intricate linguistic patterns, semantic meanings, and contextual relationships within the text makes them highly suitable for generating coherent and contextually relevant summaries. As a result, transformer models are widely used in NLP tasks, where understanding the subtleties of language is crucial to producing high-quality outputs.

3.2.3 Abstractive Summary

In this study, we employ an encoder-decoder architectural structure for abstractive text summarization. Unlike conventional approaches that rely on a single-layer encoder, the proposed technique utilizes a multi-layer encoder to enhance the model's

ability to capture complex linguistic patterns and contextual dependencies. In order to provide a consistent representation of words throughout the summarization process, the embedding layer is present in both the encoder and the decoder. Formally speaking, let w stand for a word (or token), E(w) for its matching embedding vector, and d for the size of the embedding space [87]. The decoder uses the contextualized representation of the input text that the encoder created using these embeddings to create an abstractive summary that is well-structured. In order to improve the model's ability to manage intricate sentence patterns, long-range relationships, and linguistic variances in Urdu text summarization, a multi-layer encoder is incorporated into the suggested method.

$$E(w) \in \mathbb{R}^d \tag{3.5}$$

In the model, the encoder is composed of multiple transformer layers, while the decoder utilizes a single transformer layer. The encoder consists of a stack of several transformer layers, typically ranging from 6 to 12 layers, depending on the complexity and scale of the model [88]. Each of these layers performs specific functions and contributes to the overall ability of the model to process and represent the input sequence.

These stacked transformer layers are responsible for processing the input data using self-attention mechanisms, which are central to the transformer's ability to capture complex relationships, patterns, and dependencies within the input sequence. The self-attention mechanism is what allows the model to weigh the importance of each word in the sequence relative to all other words, enabling it to dynamically adjust the focus of attention as it processes the sequence. This means that each word can "attend" to every other word, regardless of their distance from each other in the sequence. This is in contrast to traditional models like recurrent neural networks (RNNs), which process the sequence word by word and can struggle to capture long-range dependencies.

By leveraging self-attention, the model can understand the context and semantic meaning of each word in relation to the entire sentence. It can recognize subtle

nuances in meaning that depend on the surrounding words, allowing it to grasp the full meaning of the sentence or document. This contextual understanding is preserved even when the words are far apart in the sentence, making transformers particularly powerful for tasks that require deep comprehension, such as text summarization, machine translation, and question answering. The flexibility of self-attention also allows the model to process words in parallel, significantly improving efficiency and enabling the handling of long sequences of text.

The vertical stacking depicted in Figure 3.9 illustrates the presence of multiple hidden levels within the encoder, with each level corresponding to a distinct transformer layer. In this architecture, each transformer layer operates sequentially, processing the input data in a hierarchical manner. After a layer processes its input, the output is passed on to the layer above it. This allows each subsequent layer to refine and enhance the model's understanding of the input by building upon the representations generated by the previous layer.

To capture complex language patterns and nuanced connections within the input sequence, hierarchical data processing is essential. The input data is gradually transformed and refined by each encoder layer, enabling the model to create more complex and abstract representations of the underlying text. The model learns to identify not only the explicit meanings of individual words or tokens but also the intricate relationships and contextual interactions that occur between them as the data moves through many levels. Each word or token has been transformed into a highly sophisticated, contextually aware representation by the time the data reaches the encoder's last layer.

The model is able to identify meanings that go beyond superficial interpretations because of this high-level transformation, which captures the text's syntactic and semantic subtleties. The model achieves a richer understanding of language through this multi-layered approach, which enables it to digest complex phrase patterns, clear up ambiguities, and more accurately infer contextual linkages. In the end, this hierarchical encoding technique enables the model to comprehend not just discrete language components but also their more comprehensive meaning within a phrase, paragraph, or article.

The successive layers help the model gradually build a more comprehensive understanding of the sentence, allowing it to capture long-range dependencies and subtle semantic relationships. This deep, layered understanding is essential for tasks like text summarization or machine translation, where the overall meaning of the input needs to be preserved in the output. Once the final transformer layer in the encoder has processed the data, the refined representation is passed on to the next stage of processing, which may involve a decoder for generating the output, such as in tasks like sequence generation or translation.

For a given input x, a single multi-head attention layer computes the following: Q, K, and V are linear transformations of the input x, and d k is the dimensionality of the key vectors [89].

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.6}$$

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{3.7}$$

The input from the embedding layer is processed by a single transformer layer in the decoder, which then produces the output sequence [90]. Additionally, a hidden multi-head attention mechanism built into the decoder guarantees that predictions for a given token rely completely on the tokens that have come before it,as shown in figure 3.9.

The attention mechanism plays a key role in determining which hidden states, or more specifically, which segments of the encoder's output, the decoder should focus on in order to generate each word in the output sequence [91]. This allows the model to dynamically allocate attention to different parts of the input sequence based on their relevance to the current output word being generated. Rather than processing the entire input sequence equally, the attention mechanism enables the model to focus more on the parts of the input that are most important for producing the next word in the output [92]. The dot product provides a measure of similarity between the encoder's hidden states and the decoder's current state,

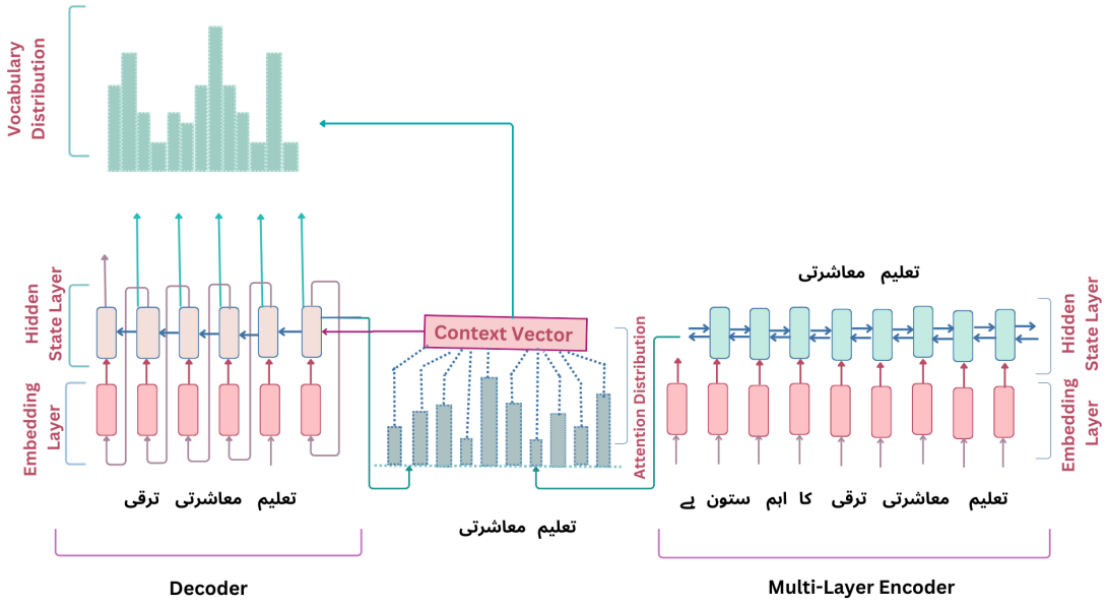

FIGURE 3.9: Multilayer Encoder Abstractive Urdu Text Summarization.

with higher values indicating a stronger relationship between the two. These attention scores are then normalized, typically using a softmax function, to produce attention weights that sum to 1, representing the relative importance of each encoder hidden state for generating the current output word.

$$\alpha_t = \text{softmax}(h_t^{\text{decoder}} \cdot h^{\text{encoder}}) \tag{3.8}$$

By leveraging the attention mechanism, the decoder is able to selectively attend to different parts of the input, enabling it to generate more accurate and contextually relevant output, especially in tasks like machine translation, text summarization, or sequence generation. The encoder's hidden states are added together and weighted to determine the context vector [93].

$$c_t = \sum_t \alpha_t h^{\text{encoder}} \tag{3.9}$$

In the suggested model, the next word in the summary is predicted using a softmax function based on the context vector and hidden state of the decoder [94]. By calculating the probability distribution throughout the vocabulary, the softmax layer makes sure that the most pertinent word is chosen at every stage of the

decoding process. This approach dynamically modifies word selection depending on previously generated tokens, enabling the model to provide summaries that are both fluent and contextually relevant. The proposed method ensures that Natural Language Processing (NLP) analysts and specialists get useful information when working on different sequence-to-sequence (seq2seq) tasks, including question-answering systems, machine translation, and text summarization [95]. The increased vocabulary distribution layer is an essential part of abstractive summarization models since it increases the resulting text's correctness, coherence, and relevance. A multilingual variant of the BART (Bidirectional and Auto-Regressive Transformers) model is called the mBART model [96]. It combines the benefits of pre-trained sequence-to-sequence models with the ability to handle many languages [97], making it helpful for tasks like translation, summarization, and paraphrase in a multilingual setting.

Figure 3.13 illustrates the working of the mBART architecture [12], where tokens are smaller units derived from the input text. Tokenizers, most frequently subword tokenizers like Byte Pair Encoding (BPE) or WordPiece, are used for this purpose. Tokenization breaks up text into smaller, more manageable chunks, allowing the model to process and understand a broader range of languages and uncommon words, as shown in Figure 3.10.

After tokenization, a dense vector representation (embedding) is assigned to each token ID. Using token embeddings, each token is converted into a numerical representation. The token embedding for each token is shown in Figure 3.11. Using the BART (Bidirectional and Auto-Regressive Transformers) architecture for abstractive text summarization, Figure 4.11 illustrates the token embedding procedure. This figure is crucial in understanding how individual tokens from the input text are transformed into numerical representations that the model can process. The process begins with individual words or subwords from the source text, which serve as input tokens. Before the text can be processed, it must first be tokenized. Each input token is mapped to a high-dimensional space by the token embedding layer, transforming it into a dense vector representation. This transformation enables the model to understand context by capturing word relationships and semantic

meanings. The embedding layer utilizes an embedding matrix, also known as a learned parameter matrix, to represent vocabulary tokens. The size of this matrix is determined by the vocabulary size and the embedding dimension. While higher-dimensional embeddings can capture more intricate relationships between tokens, they require increased computational resources. Therefore, selecting an appropriate dimensionality is a trade-off between efficiency and performance.

Positional encoding is frequently added to token embeddings in transformer-based models to offer information about each token's location within the sequence. The idea of word placement is not fundamental in transformers, in contrast to classic recurrent models that record word order. When it comes to tasks like text summarization, where the sequence of information can greatly affect meaning, positional encoding makes sure the model can preserve the text's sequential structure.

The embedding layer of the BART (Bidirectional and Auto-Regressive Transformers) model creates the final token representations, which are dense vectors enhanced with positional and semantic information. The model can better comprehend contextual links, preserve coherence, and provide fluid summaries by using these enhanced token representations, which are subsequently forwarded to other layers for additional processing. By using positional encoding, the model improves its capacity to discern between identical words that appear at various locations and successfully captures long-range relationships in the text, resulting in summaries that are more precise, contextually aware, and insightful.

A key component of the BART model's design, Figure 3.11, focuses on the token embedding procedure. It illustrates the process of transforming input tokens into numerical vectors, which is an essential stage in jobs involving natural language processing. The image highlights how embeddings are crucial for capturing the semantic links between words, which are necessary for producing summaries that are both coherent and pertinent to the context. As input to the encoder and decoder layers, the token embeddings serve as the cornerstone of the overall BART design. For natural language processing researchers and practitioners, comprehending this process is essential, particularly when working on tasks involving text summarization and other language creation applications.

Token	Token ID	Embedding Vector (128-dimensional)
پھلوں	E[0]	[0.1, 0.2, ..., 0.8]
اور	E[1]	[0.3, 0.4, ..., 0.9]
سبزیوں	E[2]	[0.5, 0.6, ..., 0.1]
کا	E[3]	[0.2, 0.4, ..., 0.3]
استعمال	E[4]	[0.7, 0.8, ..., 0.5]
انسانی	E[5]	[0.4, 0.3, ..., 0.9]
-	E[14]	[0.6, 0.5, ..., 0.8]

FIGURE 3.10: Token Embedding.

Positional embeddings are added to the token embeddings to indicate each token's position in the sequence. Positional embeddings provide the essential details on the sequence's structure, since transformers lack a core comprehension of token order [98]. The model adds positional embeddings to maintain token order, as shown in Table 3.1. Suppose we have a positional embedding matrix P of shape $(\text{max\_seq\_len}, \text{embedding\_dim})$, where the maximum sequence length is represented by max_seq_len. The Positional Embedding Vectors utilized in the BART (Bidirectional and Auto-Regressive Transformers) model for abstractive text summarization are shown in Table 3.1. In transformer architectures, positional embeddings are vital because they tell us where each token is in the input sequence, which is necessary to keep the words in a sentence in the correct order.

Transformer models utilize positional embeddings to encode information about the positions of tokens within a sequence. Since transformers do not have an inherent sense of order, unlike recurrent neural networks (RNNs) or convolutional neural networks (CNNs), positional embeddings are crucial for helping the model understand the relative position of each token in the sequence and, therefore, the contextual relationships between them. By incorporating positional embeddings, transformers are able to process the entire sequence of tokens simultaneously, while still retaining the ability to recognize word order and dependencies.

For example, in a sequence of six tokens, the positions will be labeled from 0 to 5, with each position corresponding to a specific token in the sequence. The Positional Embedding Vector column contains the positional embedding vectors

associated with each position. These vectors are learned during training and are added to the token embeddings to provide the model with information about the token's position in the sequence. The positional embedding vector for each position is designed to represent the specific location of the token within the sequence, allowing the model to distinguish between different positions and understand the relationships between tokens in context.

By combining both the token embeddings and the positional embedding vectors, transformer models are able to process the sequence holistically while still preserving the important information regarding token order and positioning. Using techniques like the sine and cosine functions, these vectors are fixed-dimensional representations of real numbers. Position 0 (0, 0.2,..., 0.8), Position 1 (0, 0.3, 0.4,..., 0.9), and Position 2 (0, 0.5, 0.6,..., 0.1) are examples of numeric values that may be shown in the table for each positional embedding vector. The dimensionality of these vectors is important since it increases the computing strain but enables more sophisticated representations and greater learning. The design of the model and the needs of the job are frequently the basis for the dimensionality selection.

Table 3.1, which focuses on the positional embedding vectors, is an essential component of the design of the BART model. The model's encoding of positional information is clearly depicted, which is essential for comprehending transformer systems. Additionally, the chart highlights how crucial it is to preserve the token order in the input sequence, which is crucial for activities like text summarization. In order to ensure that their models adequately represent the sequential nature of input data, researchers and practitioners can use it as a guide when implementing positional embeddings. Particularly for tasks like text summarization and other language creation applications, this knowledge is essential for natural language processing researchers and practitioners.

As shown in Figure 3.12, the final input embeddings are derived from the combination of both token embeddings and positional embeddings. The final embeddings for each token are computed by adding the token embeddings and positional embeddings together, allowing the model to capture both the semantic meaning of the token and its relative position within the sequence [75]. This approach enables the

Position	**Positional Embedding Vector (128-dimensional)**
0	[0.1, 0.2, . . . , 0.8]
1	[0.3, 0.4, . . . , 0.9]
2	[0.5, 0.6, . . . , 0.1]
3	[0.2, 0.4, . . . , 0.3]
4	[0.7, 0.8, . . . , 0.5]
5	[0.4, 0.3, . . . , 0.9]
14	[0.6, 0.5, . . . , 0.8]

TABLE 3.1: Positional Embedding Vectors

transformer model to effectively process input sequences while considering both the content and order of the tokens.

Token embeddings are dense vector representations of input tokens, where each token in the sequence is translated into a high-dimensional vector that encapsulates its semantic meaning. These embeddings help the model understand the underlying concepts and relationships between tokens in the text. As described in Figure 3.12, the process begins by using an embedding matrix, where each row corresponds to a unique language item (token). This matrix is learned during training, and the embeddings capture rich linguistic features that represent the meaning of each token in the context of the sentence or document.

However, transformers are inherently unaware of the order of tokens, as they process the entire sequence in parallel rather than sequentially. To address this limitation, positional embeddings are introduced. These embeddings provide information about the relative position of each token in the input sequence. By adding the positional embeddings to the token embeddings, the model is able to distinguish tokens not only by their semantic meaning but also by their position in the sequence. This positional information is essential for the transformer model to understand the contextual relationships between words, especially when the order of tokens impacts the overall meaning of the sentence or document.

Together, token and positional embeddings enable the model to process the text holistically, retaining both semantic meaning and the important contextual order of the tokens within the sequence. By combining token and positional embeddings by element-wise addition, as shown in Figure 3.12, each token's final resultant

vector will contain both its semantic meaning and its position in the sequence, and by combining this positional information with the token's intrinsic meaning, the model will have a more comprehensive understanding of the context in which each token appears. To make processing easier in later layers of the model, the final embedding will have the same dimensionality as the individual token and positional embeddings.The final embedding is then fed into the transformer layers of the BART model, which helps the model better capture the relationships between tokens and the larger context of the text, improving its ability to produce coherent and contextually relevant summaries. By combining positional and token embeddings in this way, the model guarantees that the embeddings can be processed efficiently in subsequent layers, resulting in more accurate and meaningful outputs. Figure 3.12, which shows how input tokens are converted into final embeddings, is an essential component of the BART model's design. For tasks such as summarization, it highlights the need of capturing positional context and semantic meaning. Since it is the input to the encoder and decoder layers, the Final Embedding forms the basis of the whole transformer implementation. To understand how the model works overall, one must understand this process.

For the model to handle text effectively, a complete representation of input tokens is created by combining positional and token embeddings, as shown in the image. Natural language processing academics and practitioners need to have this knowledge, particularly when working on problems requiring text summarization and other language creation applications.

$$FinalEmbedding = TokenEmbedding + PositionalEmbedding \tag{3.10}$$

The encoder processes input sequences through multiple stacked transformer layers [99]. The Self-Attention Mechanism enables the model to focus on different input parts simultaneously, capturing contextual relationships crucial for summarization. Following self-attention, Feedforward Neural Networks (FFNNs) apply non-linearity to refine the output, enhancing representation learning [100]. This combination of transformer layers, self-attention, and FFNNs ensures the model effectively understands context, improving summarization accuracy and coherence.

Token	Position	Final Embedding (128-dimensional)
پھلوں	0	[0.2, 0.4, ..., 1.6]
اور	1	[0.6, 0.8, ..., 1.8]
سبزیوں	2	[1.0, 1.2, ..., 0.2]
کا	3	[0.4, 0.8, ..., 0.6]
استعمال	4	[1.4, 1.6, ..., 1.0]
انسانی	5	[0.8, 0.6, ..., 1.8]
-	14	[1.2, 1.0, ..., 1.6]

FIGURE 3.11: Final Embedding.

Based on the output of the encoder and the tokens created previously, the decoder creates the output sequence [101]. By utilizing masked self-attention, the decoder makes sure it only depends on earlier tokens and generates tokens in a sequential fashion. While creating each token, it uses cross-attention to concentrate on pertinent encoder outputs. The decoder is first set up with a sequence length of one and a start token [102].

The decoder layers create the summary sequence in an autoregressive fashion, one token at a time, using the encoder's output and the previously generated token as input at each decoding step. The decoder then uses this information to create a probability distribution over the entire vocabulary, predicting the most likely next token in the sequence [12]. This process is repeated iteratively, adding each new token to the growing summary until the model reaches a stopping criterion, like the end-of-sequence token. The autoregressive nature of this process keeps the model coherent and guarantees that each token in the summary is contextually appropriate based on the input text and the sequence of previously generated tokens.

Figure 3.11 illustrates the workings of Bart architecture. Two primary sentences comprise the input document's tokens: Sentence 1,CLS sent one SEP,Sentence 2,CLS sent two SEP. A separation token (SEP, indicates the end of a sentence, and a classification token (CLS, indicates the beginning of a sentence. A numerical representation is incorporated with each token.To represent input tokens, the

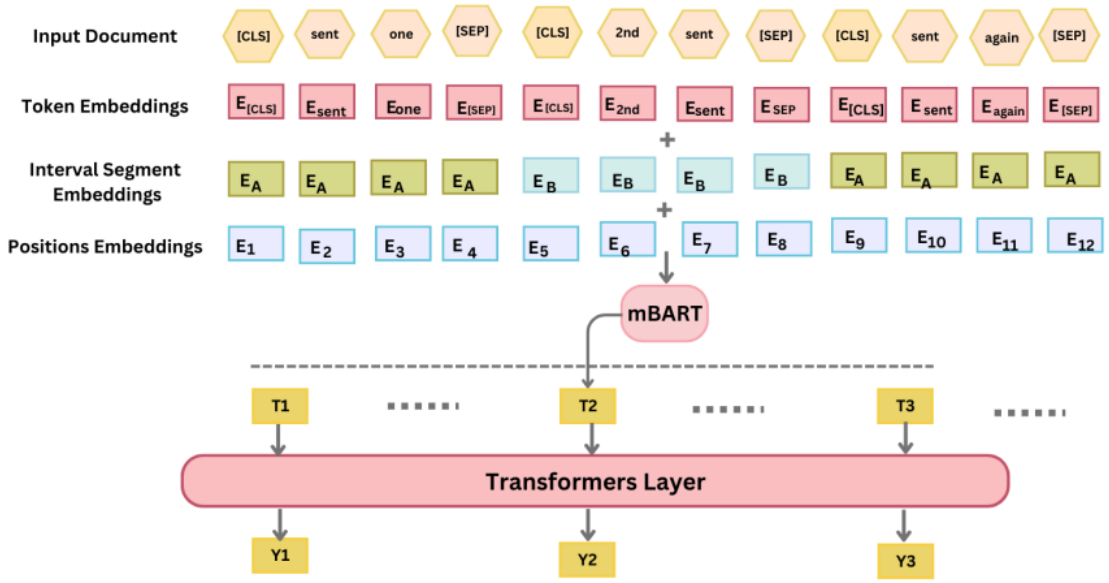

FIGURE 3.12: Bart Architecture.

model employs three different kinds of embeddings. Token, position, and interval segment embeddings are examples. Every token in the input document is transformed into a distinct vector that serves as a representation of its identity. The input's sentences and segments are each given interval segment embeddings, which are repeated for every token in each segment. The segmentation aids the model in differentiating between sentences or portions. Position embeddings use positional information to determine the sequence's token order. In addition to the matching token and segment embeddings, each token position is represented by a position embedding. Token embedding, interval segment embedding, and position embedding work together to provide a composite embedding vector that contains details about the token, its phrase or segment, and its position within the sequence. This vector is then utilized in the mBART model.

The Multilingual BART (mBART) model, a transformer-based model for sequence-to-sequence tasks like text synthesis and machine translation, receives the combined embeddings for each token. Tasks needing cross-lingual comprehension can benefit from mBART's ability to handle multilingual inputs. mBART processes the embeddings through a series of Transformer layers that use feed-forward operations and self-attention to record intricate linkages and dependencies in the token sequence. These layers take into account the input document's whole context to

improve each token's representation. The tokens $(T_1, T_2, T_3, \dots)$ for each point in the sequence represent the output from the transformer layer. Here, each token is a modified variant of the matching input token that has been enhanced with context from the full sequence. Finally, an output prediction is mapped to each changed token in the sequence $(T_1, T_2, T_3, \dots)$ $(Y_1, Y_2, Y_3, \dots)$. Words in a translated sentence, labels in a classification task, or any other output pertinent to the particular application could be represented by these.

In order to comprehend the BART model architecture and how it is used in the suggested abstractive text summarization approach for Urdu, Figure 3.13 is an essential visual help. It shows the architecture of the model graphically, emphasizing important elements like output creation and tokenization. Additionally, the graphic demonstrates the usage of transformer architecture, which captures contextual relationships within the text using self-attention techniques. For natural language processing academics and practitioners to produce high-quality summaries, especially in low-resource languages like Urdu, this knowledge is crucial. In general, Figure 3.13 is crucial for natural language processing practitioners and academics.

Chapter 4

Experiment and Results

4.1 Introduction

In this section, a series of tests is used to assess the efficacy of the suggested summarizing technique. In Section A, data sets are described. Section B provides an explanation of the evaluation measures. The setup of the experiments is covered in Section C. In section D, the experiments, findings, and discussion are presented. Section E concludes with a comparison with other comparable systems.

4.1.1 Dataset

Sports, Science, Technology, Business, Economics, and Entertainment are the four categories into which the 1,000 news data sets used for this study are divided. It consists of long news stories and their abridged summaries. The model was trained using 80% of the data, with the remaining 20% being utilized for testing. It is an enormous Urdu dataset accessible for solving natural language processing-related challenges. Table 4.1 displays the information gathered from multiple sources.

The study's data sources for creating the abstractive text summarization model in Urdu are listed in Table 5.1. Understanding the size and variety of the datasets used is essential, since this table can have a big influence on the summarization

TABLE 4.1: Data Collections from Various Sources.

S.No	Source	Total Sentences
1	Chetor	166312
2	Wikipedia	1878008
3	Digikala	177357
4	BigBang Pag	3017
5	Books	25335

model's performance and generalization. The S.No (Serial Number) column in Table 5.1 provides a sequential number for each data source, which helps to organize the information and provides convenient access. The names of the data sources from which the text data was gathered—such as news articles, blogs, or other written materials pertaining to the Urdu language—are included in the source column. A more robust dataset is indicated by a higher number of sentences in the 'Total Sentences' column, which shows the total number of sentences in each data source. This information is essential for evaluating the amount of data available for summary model training and testing.

A content source, a Wikipedia source, a Digikala source, a Big Bang Pag source, and a book source provide the dataset for a model based on Urdu content. The content source most likely alludes to a particular website or platform that offers 166,312 sentences of Urdu material. The 1,878,008 sentences in the Wikipedia source demonstrate a wide variety of subjects and writing styles. The dataset's richness is enhanced by the 177,357 phrases found on the Iranian e-commerce portal Digikala. Despite having just 3,017 phrases, the smaller site Big Bang Pag could nevertheless offer original material. The books category contains 25,335 sentences of diverse Urdu literary or educational resources. By combining formal and organized writing styles, the books category deepens the dataset.

The variety of datasets employed, the amount of data available for training, and the model's contextual relevance are all highlighted in Table 4.1, which is essential for building a strong summarization model. A diverse dataset aids in the model's ability to efficiently summarize various kinds of material. Each source's total amount of words offers insight into the data that is accessible, and including sources

such as Wikipedia and e-commerce platforms improves the model's capacity to produce pertinent summaries in a variety of scenarios.

4.1.2 Evaluation Measures

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measure is used in the quantitative evaluation in addition to the metrics recommended in this study [103]. It comprises a set of measures for automatic assessment of text summarization. The results show that dual and multiple encoder models outperform single encoder models in terms of performance. ROUGE1, ROUGE2, ROUGE-L, and BLEU score measures are used to assess the final summary [104]. These metrics are calculated using the following formulae [70].

ROUGE-1

$$\text{Precision (P)} = \frac{\text{Number of Overlapping Unigrams}}{\text{Total words in Automatic Summary}} \tag{4.1}$$

$$\text{Recall (R)} = \frac{\text{Number of Overlapping Unigrams}}{\text{Total words in Reference Summary}} \tag{4.2}$$

$$\text{F1} = \frac{2(P \times R)}{P + R} \tag{4.3}$$

ROUGE-2

$$\text{Precision (P)} = \frac{\text{Number of Overlapping Bigrams}}{\text{Total words in Automatic Summary}} \tag{4.4}$$

$$\text{Recall (R)} = \frac{\text{Number of Overlapping Bigrams}}{\text{Total words in Reference Summary}} \tag{4.5}$$

$$\text{F1} = \frac{2(P \times R)}{P + R} \tag{4.6}$$

ROUGE-L

$$\text{LCS} = \max\{s_1, s_2, s_3, \ldots, s_n\} \tag{4.7}$$

$$\text{Precision (P)} = \frac{|LCS|}{\text{Total words in Automatic Summary}} \tag{4.8}$$

$$\text{Recall (R)} = \frac{|LCS|}{\text{Total words in Reference Summary}} \tag{4.9}$$

$$\text{F1} = \frac{2(P \times R)}{P + R} \tag{4.10}$$

Using the BLEU score (BiLingual Evaluation Understudy), the model's performance is further assessed [68].The ROUGE score is mirrored in the BLEU score. Its main application is in machine translation jobs, while summarization tasks have also been found to be evaluated with it [71]. To get the BLEU score, one needs to calculate the number of n-grams in the candidate sentence that match the reference phrase.The following equation represents the shortness penalty (BP), n-gram precisions (pn), n-gram length (N), and positive weights (wn) [72].

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{4.11}$$

As an evaluation measure, we employ the BERTScore [69], since the ROUGE and BLEU scores are primarily applied to extractive text summarization and do not take implicit semantics the use of different words for the same meaning into consideration [104]. BERTScore is dependent on the use of BERT contextual

embeddings. Instead of calculating the matching between the candidate and reference sentences, the BERTScore determines the level of token similarity [69]. As the BERTScore is similar to assessments made by humans, it is a good score for our assignment. This formula can be seen below. For the BERTScore, FBERT stands for the F-measure, RBERT for recall, and PBERT for accuracy [74].

$$F_{\text{BERT}} = \frac{2 \cdot (P_{\text{BERT}} \times R_{\text{BERT}})}{P_{\text{BERT}} + R_{\text{BERT}}} \tag{4.12}$$

4.1.3 Experiments setup and Tools

For the purpose of implementing various tasks, including sentence representation and feature extraction, we mostly utilized Java. Because the tools we chose are Java-based and are published as JAR files, we also utilized Java to integrate the tools. Table 4.2 provides an overview of the strategies, resources, and constraints used in the recommended approach.

4.1.4 Obtained Results

As an illustration, the appropriate range for the summary extent ratio is 33–40%; yet, some summaries have a size ratio of as much as 80% of the provided information.Approximately thirty items are categorized into several sections, as depicted in Figure 5.1. It is never easy to get a computer to match human language, natural language processing skills, and lexicon perfectly.

There were 10–21 lines of Urdu text in the evaluation paper. Figure 4.2 shows how the five lines, or about half of the entire article, that make up the sentence weight technique used to generate the summary. Sentence weight and the TF-IDF approach were used to group and categorize the most significant sentences. Consequently, the resulting summary preserved the main points of the original text, consisting of 5–6 lines, or approximately 51% of the original document seen in Figure 4.3. A combination of meaningful and arbitrary sentences were seen in the produced summary. Selected the longest, most wordy sentences. After

then, this summary is used to construct the abstractive summary, which makes up half of the extractive summary. It is necessary for the encoder-decoder model to employ the pre-trained model mBART in order to generate the abstractive summary. Table 4.2 describes the parameters that are employed in our system and the execution time of each step is illustrated in Table 4.3.

Table 4.2: Variable Parameters in Our System

Tasks/Tools	**Our Usage**
Preprocessing tools	Pandas: Data loading and saving.
	Regex: Tokenization, punctuation removal, and formatting.
	Scikit-learn (TfidfVectorizer): Text feature extraction to compute TF-IDF scores.
	NumPy: Numerical operations for scoring sentences.
	NLTK: BLEU score calculation for evaluation.
	Transformers (mBART): Tokenization and abstractive summarization generation.
Key-Phrase extraction	TF-IDF for scoring sentence importance.
Sentence representation	Bag-of-words model with TF-IDF scores.
ROUGE tool	ROUGE for evaluation metrics (precision, recall, F1-score) of generated summaries against original text.

The variable parameters utilized in the suggested abstractive text summarizing system are summarized in Table 4.2. This table is crucial for comprehending the instruments, methods, and setups that affect the model's functionality. Each stage in the model's workflow corresponds to a particular job or tool used in the summarization process. The use part of the suggested system describes how each activity or tool is implemented and offers insights into the technology and approaches employed to meet the summary goals. Pandas is a data manipulation Python package that is used to import and store structured data.

Tokenization, punctuation removal, and formatting are all done via regular expressions. Text features are extracted and TF-IDF scores are computed using Scikit-learn (TfidfVectorizer). For scientific computing, NumPy is a Python module that supports huge, multi-dimensional arrays and matrices. To assess the quality of machine-translated text, NLTK is utilized to compute the BLEU score for evaluation. These technologies are necessary for jobs involving text processing,

as well as for recognizing and working with certain textual patterns.By comparing the frequency of important terms across documents, the TF-IDF approach is utilized to detect them. Frequently utilized in text categorization and summarizing applications, the Bag-of-Words Model with TF-IDF Scores captures word frequency without disregarding word order. Focusing on the overlap of n-grams between the generated and reference summaries, the ROUGE Tool compares the precision, recall, and F1-score of created summaries to the source text. Table 4.2 is essential because it provides performance insights, repeatability, and technique clarity. It helps readers comprehend the technical underpinnings of the suggested model by offering a thorough explanation of the instruments and methods employed in the summarizing process. The table also demonstrates the thorough strategy used in the study, integrating many methods to improve the quality of the summaries.

The abstractive summary showcases the capability of the proposed system to generate translations that incorporate a rich and varied vocabulary when processing Urdu phrases. This approach demonstrates an advanced level of linguistic flexibility, allowing the system to go beyond mere word-for-word translation and instead produce coherent and contextually relevant summaries.

Unlike extractive summarization, which retains sentences or phrases directly from the source text without altering their structure, the abstractive summarization technique synthesizes new sentences that preserve the original meaning while enhancing readability and fluency. This process involves paraphrasing, restructuring, and integrating alternative vocabulary choices, ensuring that the generated summaries are not only concise and accurate but also linguistically diverse and contextually appropriate.

Translation quality, accuracy, and contextual relevance are all improved by the system's careful design, which makes use of sophisticated natural language processing (NLP) techniques.

This guarantees that the output is not only linguistically accurate but also fluid, natural, and flexible enough to accommodate a variety of use cases. In addition

to enhancing readability and coherence, the system skillfully rewords and restructures text while preserving the original idea. This feature is especially helpful in situations when literal translations could miss the little details, cultural allusions, and deeper meanings included in the source Urdu text. The inability of traditional translation techniques to maintain idiomatic idioms, tone, and contextual meanings might result in misunderstandings. The suggested approach cleverly fills this gap by using context-aware summary techniques, which improve accessibility and clarity while maintaining the translated summaries' integrity to the original material. The technique greatly improves comprehensibility and expressiveness using this advanced method, making the condensed information more interesting, educational, and available to a wider range of users. This enhances the user experience while also promoting more successful cross-cultural and multilingual communication. The generated abstractive summary is intentionally crafted to be concise and compact, effectively retaining the essential information from the original document while eliminating unnecessary redundancy. This summarization method focuses on distilling the most critical points, ensuring that the resulting summary is both informative and easy to digest. As shown in Figure 4.1, the summary constitutes approximately 20% of the original test document, which showcases the system's impressive ability to compress the text without sacrificing key insights.

This level of compression highlights the efficiency of the abstractive summarization process, where the core ideas are maintained, and less relevant details are discarded. The system's effectiveness in distilling key points while minimizing excess is a testament to its performance.

Additionally, Figure 4.4 provides a detailed comparative analysis between the number of lines in the original raw text and those in the generated summary. This visual representation offers valuable insights into the system's compression capabilities, demonstrating how it reduces the length of the document while preserving its core meaning. The analysis emphasizes the system's ability to condense large volumes of text into more manageable summaries, making it a powerful tool for information extraction and communication. This approach ensures that the summaries remain informative while significantly reducing the length of the original

text, making it more accessible and easier to comprehend.

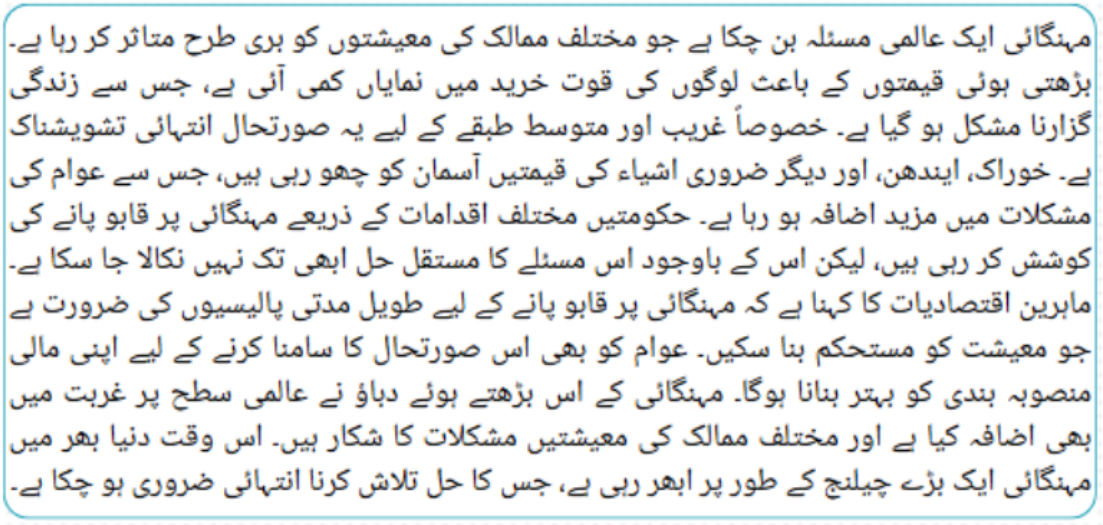
مہنگائی ایک عالمی مسئلہ بن چکا ہے جو مختلف ممالک کی معیشتوں کو بری طرح متاثر کر رہا ہے۔ بڑھتی ہوئی قیمتوں کے باعث لوگوں کی قوت خرید میں نمایاں کمی آئی ہے، جس سے زندگی گزارنا مشکل ہو گیا ہے۔ خصوصاً غریب اور متوسط طبقے کے لیے یہ صورتحال انتہائی تشویشناک ہے۔ خوراک، ایندھن، اور دیگر ضروری اشیاء کی قیمتیں آسمان کو چھو رہی ہیں، جس سے عوام کی مشکلات میں مزید اضافہ ہو رہا ہے۔ حکومتیں مختلف اقدامات کے ذریعے مہنگائی پر قابو پانے کی کوشش کر رہی ہیں، لیکن اس کے باوجود اس مسئلے کا مستقل حل ابھی تک نہیں نکالا جا سکا ہے۔ ماہرین اقتصادیات کا کہنا ہے کہ مہنگائی پر قابو پانے کے لیے طویل مدتی پالیسیوں کی ضرورت ہے جو معیشت کو مستحکم بنا سکیں۔ عوام کو بھی اس صورتحال کا سامنا کرنے کے لیے اپنی مالی منصوبہ بندی کو بہتر بنانا ہوگا۔ مہنگائی کے اس بڑھتے ہوئے دباؤ نے عالمی سطح پر غربت میں بھی اضافہ کیا ہے اور مختلف ممالک کی معیشتیں مشکلات کا شکار ہیں۔ اس وقت دنیا بھر میں مہنگائی ایک بڑے چیلنج کے طور پر ابھر رہی ہے، جس کا حل تلاش کرنا انتہائی ضروری ہو چکا ہے۔

FIGURE 4.1: Input Source Test Document.

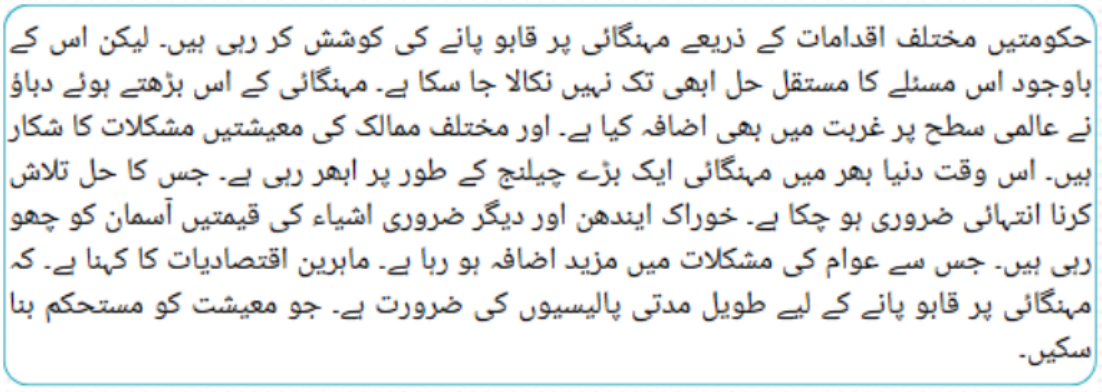
حکومتیں مختلف اقدامات کے ذریعے مہنگائی پر قابو پانے کی کوشش کر رہی ہیں۔ لیکن اس کے باوجود اس مسئلے کا مستقل حل ابھی تک نہیں نکالا جا سکا ہے۔ مہنگائی کے اس بڑھتے ہوئے دباؤ نے عالمی سطح پر غربت میں بھی اضافہ کیا ہے۔ اور مختلف ممالک کی معیشتیں مشکلات کا شکار ہیں۔ اس وقت دنیا بھر میں مہنگائی ایک بڑے چیلنج کے طور پر ابھر رہی ہے۔ جس کا حل تلاش کرنا انتہائی ضروری ہو چکا ہے۔ خوراک ایندھن اور دیگر ضروری اشیاء کی قیمتیں آسمان کو چھو رہی ہیں۔ جس سے عوام کی مشکلات میں مزید اضافہ ہو رہا ہے۔ ماہرین اقتصادیات کا کہنا ہے۔ کہ مہنگائی پر قابو پانے کے لیے طویل مدتی پالیسیوں کی ضرورت ہے۔ جو معیشت کو مستحکم بنا سکیں۔

FIGURE 4.2: Extractive Summary from the Input Source Document.

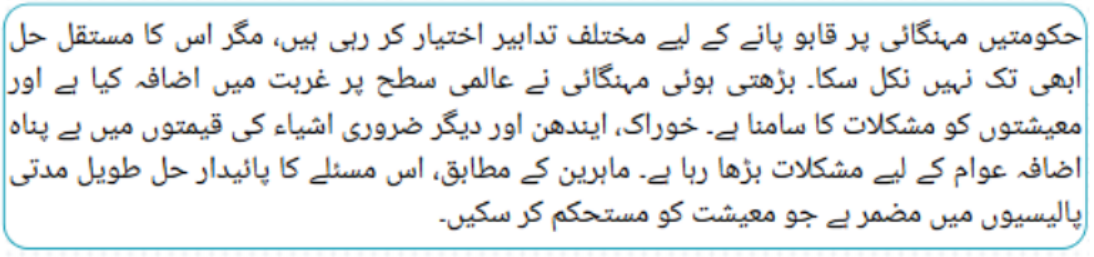
حکومتیں مہنگائی پر قابو پانے کے لیے مختلف تدابیر اختیار کر رہی ہیں، مگر اس کا مستقل حل ابھی تک نہیں نکل سکا۔ بڑھتی ہوئی مہنگائی نے عالمی سطح پر غربت میں اضافہ کیا ہے اور معیشتوں کو مشکلات کا سامنا ہے۔ خوراک، ایندھن اور دیگر ضروری اشیاء کی قیمتوں میں بے پناہ اضافہ عوام کے لیے مشکلات بڑھا رہا ہے۔ ماہرین کے مطابق، اس مسئلے کا پائیدار حل طویل مدتی پالیسیوں میں مضمر ہے جو معیشت کو مستحکم کر سکیں۔

FIGURE 4.3: Abstractive Summary from Extractive Summary.

Figure 4.4 shows the comparison of the number of lines in the output summary produced by the summarization model with the number of lines in the raw text. To comprehend how well the summary method reduces long texts into digestible

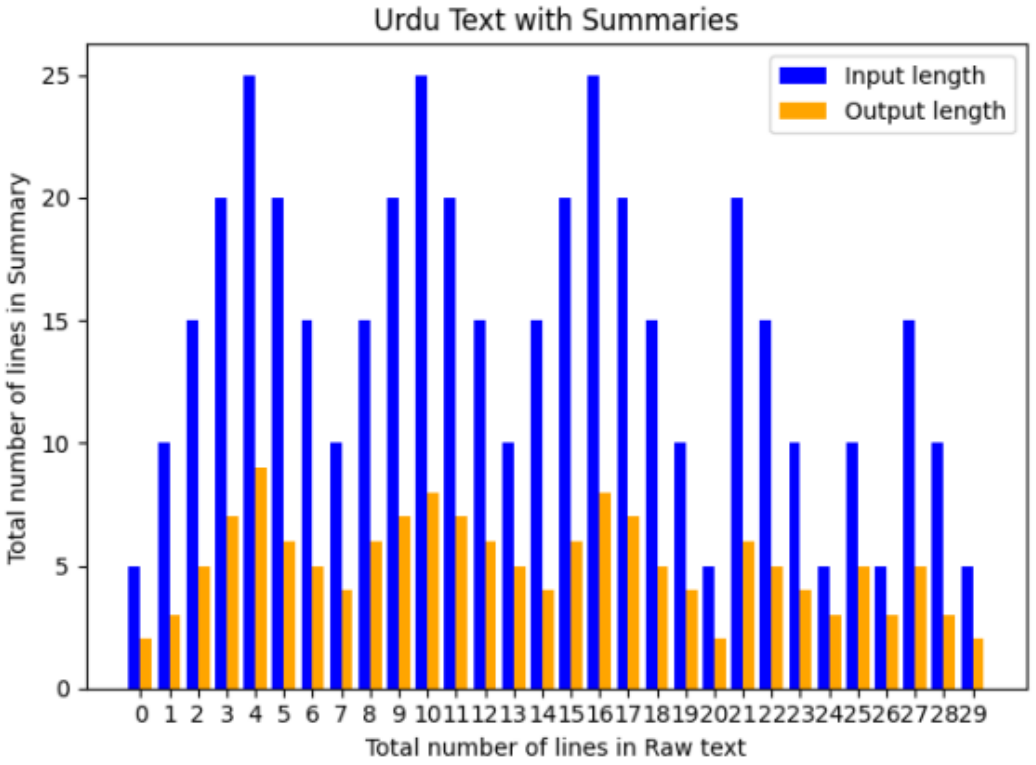

FIGURE 4.4: Analysis of the Raw Text's Line Count and Comparison of that Number to the Output Summary.

summaries, this graphic is essential. The summarizing model's capacity to condense text while maintaining important information is graphically demonstrated in the figure, which also emphasizes the model's capacity to extract important elements and exclude unnecessary ones.

It is composed of up of two parts: the Output Summary Line Count, which shows how much condensed text was included in the model's summary, and the Raw Text Line Count, which shows how many lines there were in the original document overall.

Effective summarizing is demonstrated by the figure, which most likely has a bar graph. It illustrates a notable decrease in the number of lines from raw text to output summary. The model successfully condensed the material to its most important components if the original text is 20 lines long and the summary is 5 lines long. According to the study, the summarizing approach improves readability and comprehension for users who might not have the time to read the complete article by efficiently identifying and retaining the major concepts while eliminating

less important material. This implies that the summarizing method preserves the integrity of the original material, guaranteeing that key points are communicated without suffering a substantial loss of meaning. A key aspect in applications such as news stories, reports, and academic papers that need fast information retrieval is the summarizing model's capacity to effectively condense long texts into brief summaries, as seen in Figure 4.4, which visually confirms the model's performance.

The typical execution time for each job in the suggested abstractive text summarization model is shown in Table 4.3. Model complexity refers to the time taken to initialize and use a model. It involves several steps, including tokenizer initialization, model setup, generating summaries, and calculating BERTScore. Tokenizer initialization takes 4.38 seconds, which is usually independent of input size n. Model setup takes 0.7 seconds, which is also O(1). Generating summaries takes 53.9 seconds, which is the most time-consuming step and directly scales with input size n.

The complexity depends on the size of the input n and the internal architecture of the model. For example, transformer-based models like BERT and GPT typically have a complexity of $O(n^2 \cdot d)$ for self-attention, where d is the embedding dimension. For smaller models, this might reduce to $O(n^2)$ for per-token processing cost. The dominant cost appears to be $O(n)$ since the model complexity is fixed for each token/sentence.

Calculating BERTScore takes 9.06 seconds, which involves comparing generated summaries to reference texts using BERT embeddings. The summary generation step is the most significant contributor to runtime in the model complexity category, with a complexity of $O(n)$ under typical use cases. If using a more complex model architecture, this could increase to $O(n^2)$ for large inputs.

Tokenization is efficient since it only takes 0.02 seconds and stores processed phrases for later use. Moreover, importing punctuation removal data is a step in the process, which entails creating the rules required for text removal. Usually, the procedure takes about 0.02 seconds to complete. Removing punctuation is an essential preprocessing step that improves text analysis; it usually takes less than

0.0 seconds. After that, phrases are cleaned and stored, which might take 0.03 seconds.

In order to determine the relevance of words in a text, the TF-IDF Vectorization phase computes Term Frequency-Inverse Document Frequency (TF-IDF) scores for tokens. This process saves scored phrases, which might take 0.03 seconds, and usually takes 0.01 seconds. This phase usually has a short execution time. The procedure consists of two stages: first, the Top N Sentences are chosen based on prior score, which usually takes 0.05 seconds. Next, the final summary is generated using the sentences that were chosen, which usually takes 0.04 seconds to complete. Compared to other processes, such loading sample text, which usually executes in 0.01 seconds, tokenizer setup is a resource-intensive procedure that might take longer. This suggests that the tokenizer's setup requires additional resources. Setting up the summarization model involves initializing and preparing it for execution, a process that typically takes 0.7 seconds. This setup phase ensures that all necessary components, such as the model's architecture, parameters, and dependencies, are correctly loaded before summarization begins.The most computationally intensive stage is the summary generation process, which requires substantial processing power. On average, this phase takes 53.9 seconds, reflecting the complexity of abstractive summarization techniques that involve deep learning models. These models analyze, understand, and rephrase content to generate coherent and contextually accurate summaries, which accounts for the significant processing time.

Once the summary is generated, the output is stored within 0.01 seconds, showcasing the efficiency of the storage mechanism. Despite the high processing demands of summary generation, the overall execution time remains relatively low, demonstrating the model's optimized performance. Additionally, the BERTScore metric, which is used to evaluate the semantic similarity between the generated and reference summaries, can be computationally demanding. This evaluation method leverages deep contextualized embeddings to assess meaning alignment, further contributing to processing overhead. In total, the entire summarization workflow, including setup, summary generation, storage, and evaluation, completes within

59.31 seconds. This time frame highlights the overall efficiency of the system while acknowledging the computational challenges associated with high-quality abstractive summarization and evaluation. An extractive summary of the execution timings for the many activities included in the abstractive text summarization model is given in Table 4.3. The table illustrates the model's efficiency and indicates the most computationally demanding jobs by segmenting the time required for each step. The considerable amount of time required to compute BERTScore and provide summaries indicates that these are important topics for improvement in further research.

TABLE 4.3: Proposed Model Accuracy.

Metric	**Precision**	**Recall**	**F-Measure**
ROUGE-1	98%	64%	78%
ROUGE-2	97%	55%	70%
ROUGE-L	98%	64%	78%
BERTScore	90%	90%	90%

The performance of the suggested abstractive text summarization model was assessed using the measures listed in Table 4.4. A mix of conventional and contemporary criteria, such as BERTScore and ROUGE scores, are commonly used to assess the caliber of the generated summaries. From superficial overlap to more profound semantic comprehension, these metrics provide a thorough examination of several facets of the summaries. ROUGE-1 measures the overlap of unigrams (single words) between the generated summary and the reference summary, indicating the content coverage; ROUGE-2 measures the overlap of bigrams (pairs of words), evaluating the preservation of word combinations and contextual relationships; ROUGE-L considers the longest common subsequence (LCS), reflecting the alignment of sentence structure between the generated and reference summaries; and BERTScore, a more recent and sophisticated metric, evaluates the semantic similarity between the generated and reference summaries by leveraging contextual embeddings from the BERT model, providing a more nuanced view of how well the summaries capture the underlying meaning. When taken as a whole, these metrics give a comprehensive picture of how well the model represents the text's surface-level and semantic elements, revealing information about its syntax

and context comprehension. Lexical overlap and semantic similarity metrics are used to provide a reliable assessment of the summarization quality. The scores for each measure are shown in the precision column. High precision is indicated by dividing the number of overlapping elements (bigrams, LCS, or unigrams) by the total number of items in the summary. Recall scores for each metric are shown in the column. These are determined by dividing the total number of reference summary items by the number of overlapping elements.

A high recall rate means that the summary successfully conveys pertinent information from the source material. A thorough understanding of the model's performance is provided by the F-measure column, which shows the harmonic mean of accuracy and recall. Better overall summary generation that balances recall and accuracy is indicated by a higher F-measure.

The remarkable accuracy and recall of the ROUGE-1 model, as evidenced by its 78% F-measure, raises the possibility that some crucial information from reference summaries is absent. With a 70% F-measure, the ROUGE-2 model's accuracy and recall are somewhat worse, indicating that it is less successful at collecting bigrams, which are essential for preserving contextual meaning in summaries.

Effective coherence is indicated by the excellent accuracy and recall of the ROUGE-L model, which can replicate the ROUGE-1 scores. The model consistently maintains the structure and flow of the generated summaries. The model captures lexical material and preserves semantic meaning, as evidenced by the BERTScore performance, which may exhibit high accuracy, recall, and F-measure (e.g., 90%), making the summaries more contextually relevant and human-like.

Table 4.4 provides a thorough and detailed summary of the evaluation metrics used to assess the performance of the proposed abstractive text summarization model. It includes an extensive set of key performance metrics, such as accuracy, recall, and F-measure, which serve as critical indicators of the model's overall effectiveness. These metrics are evaluated across several important summary evaluation measures, including ROUGE-1, ROUGE-2, ROUGE-L, and BERTScore, each contributing unique insights into different aspects of the model's performance.

The ROUGE metrics, which assess n-gram overlap (ROUGE-1 for unigrams, ROUGE-2 for bigrams, and ROUGE-L for the longest common subsequence), allow for an in-depth evaluation of how well the model maintains content accuracy and structural integrity in its generated summaries. BERTScore, on the other hand, evaluates the semantic alignment between the generated summary and the original text by leveraging contextualized BERT embeddings, offering a deeper understanding of the model's ability to preserve meaning. Together, these metrics provide a holistic view of the model's strengths and weaknesses, comparing the generated summaries with reference summaries to reveal how closely they align in terms of both lexical content and semantic accuracy.

The inclusion of ROUGE metrics, which assess the overlap of n-grams, word sequences, and word pairs, provides a valuable indication of how effectively the model captures the surface-level lexical content of the original text. ROUGE-1 focuses on unigram overlap, ROUGE-2 on bigram overlap, and ROUGE-L on the longest common subsequence, each highlighting the model's ability to preserve content coherence and integrity at different granular levels. Additionally, BERTScore, which evaluates the semantic similarity between the generated summary and the original text by utilizing BERT embeddings, further underscores the model's proficiency in producing summaries that are not only lexically rich but also semantically accurate.

The results presented in Table 4.4 highlight the effectiveness of the proposed model in the field of Urdu text summarization. The high scores achieved in both ROUGE and BERT metrics demonstrate that the model excels at generating summaries that are closely aligned with the original content, not only at a lexical level (by capturing exact word overlaps) but also at a semantic level (by preserving the overall meaning and context). These promising results emphasize the model's potential to produce high-quality summaries in Urdu, positioning it as a valuable tool for real-world applications where both accuracy and coherence are crucial.

A visual comparison of the assessment findings for the suggested method of abstractive text summarization, especially employing the ROUGE metrics (ROUGE-1, ROUGE-2, and ROUGE-L), is shown in Figure 4.5. Three important ROUGE

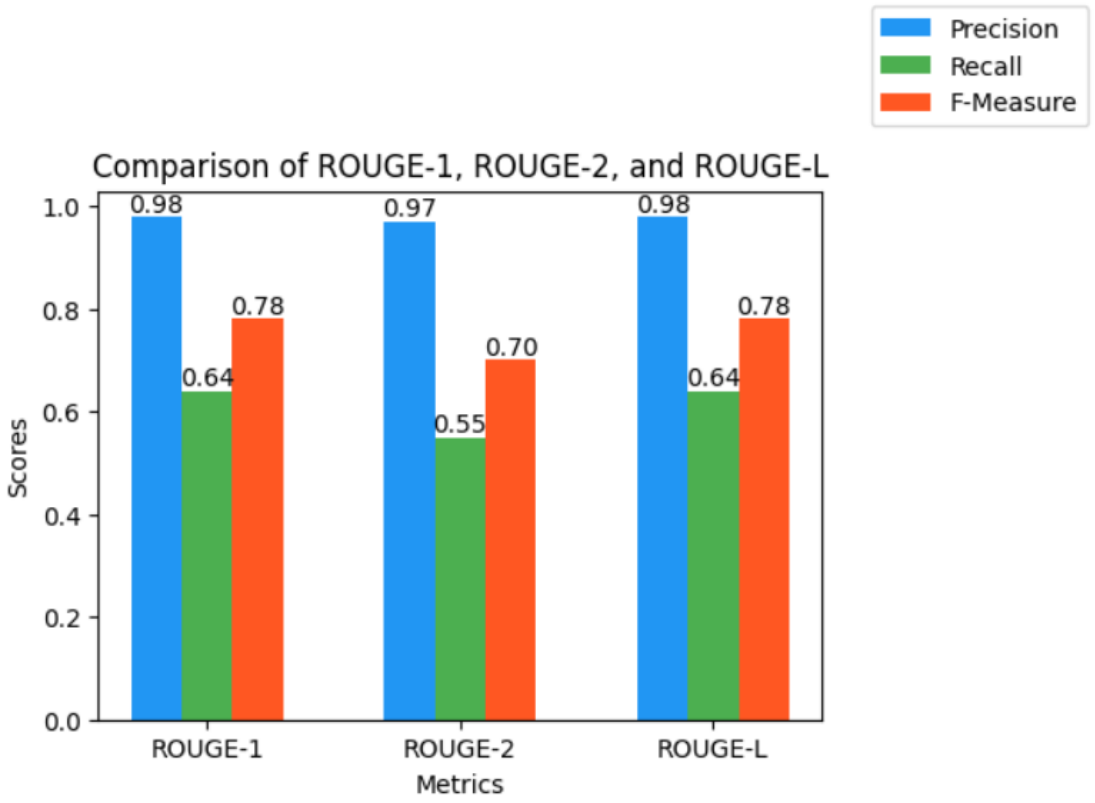

FIGURE 4.5: Proposed Approach Evaluation Results using Rouge-1, Rouge2 and Rouge L Measures

metrics are covered in the figure: ROUGE-1, which assesses lexical similarity by measuring the overlap of unigrams between generated and reference summaries; ROUGE-2, which assesses the model's ability to capture phrases and contextual meaning; and ROUGE-L, which assesses the generated text's fluency and coherence by evaluating the longest common subsequence (LCS) between generated and reference summaries.

The scores for each ROUGE indicator are shown graphically in the image, most commonly using a bar graph style, making it simple to compare the performance of the suggested approach to that of alternative approaches or baseline models. In order to illustrate the efficacy of the suggested approach relative to alternative strategies, the graphic may include bars that compare the performance of the suggested method with that of the current summarizing techniques.The suggested approach produces summaries that are lexically and structurally comparable to the reference summaries, as evidenced by its high scores on all three ROUGE measures. A substantial percentage of the produced summary matches the reference summary,

as shown by a ROUGE-1 score of 80.3%. The proposed method outperforms many existing methods, especially in ROUGE-1 and ROUGE-L scores, demonstrating its ability to capture essential text content while maintaining coherence and flow in the summary. Trends, such as the method's consistency across measurements or regions where it succeeds or struggles, may be quickly seen thanks to the visual display of ROUGE ratings. High ROUGE scores, especially in ROUGE-1 and ROUGE-L, show that the approach can generate high-quality summaries, which are important in applications where it's critical to comprehend the major concepts and preserve the structure of the original text.

4.1.5 Comparing with other Related Works

Using quantitative evaluation metrics, Table 4.5 provides an objective comparison of our proposed approach against related methodologies across both datasets. The table presents a detailed analysis of performance indicators, allowing for a comprehensive assessment of the strengths and weaknesses of each approach. Additionally, the table highlights the relative improvements achieved by our method, showcasing enhancements in key performance metrics. By leveraging these rough measures, we can effectively quantify the effectiveness of our strategy in contrast to existing techniques, offering valuable insights into its advantages and potential areas for further optimization.

Table 4.5 shows how well different authors' approaches do when it comes to text summarization utilizing rough metrics, such as rough-1, rough-2, and rough-L. With scores of 80.3 for ROUGE-1, 74.3 for ROUGE-2, and 80.3 for ROUGE-L, the suggested strategy performs well and outperforms other approaches by a wide margin, resulting in an average of 78.3. Ramesh Nallapati (2019) and Peter J. Liu (2022) both attain comparable scores with averages of about 28.7, whereas Kuan-Yu Chen (2024) achieves 47.2 (ROUGE-1), 21.5 (ROUGE-2), and an average of 32.9. With averages of 32.6 and 36.5, respectively, Rishabh Katna (2023) and Chenguang Zhu (2023) perform better than Sebastian Gehrmann (2023), who performs marginally worse. The average score for Asmaa Elsaid (2024) is better

TABLE 4.4: Comparison of the Suggested Method using the CNN/Daily Mail Dataset with other Approaches.

Authors	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
Proposed Method	80.3	74.3	80.3
Kuan-Yu Chen, 2024 [51]	47.2	21.5	30.1
Ramesh Nallapati, 2019 [52]	35.1	18.4	32.7
Peter J. Liu, 2022 [53]	35.1	18.4	37.2
Sebastian Gehrmann, 2023 [54]	35.1	18.1	32.8
Rishabh Katna, 2023 [55]	40.2	22.1	35.8
Chenguang Zhu, 2023 [56]	50.2	22.4	36.9
Jinge Yao, 2023 [57]	32.5	34.5	36.5
Asmaa Elsaid, 2024 [60]	45.7	35.6	40.6
Muhammad Aslam, 2023 [61]	87.2	82.1	85.2
Hassan Raza, 2023 [62]	35.2	20.5	30.5
Muhammad Awais, 2023 [63]	35.5	25.3	30.5

at 40.6, compared to 34.5 for Jinge Yao (2023). Muhammad Aslam (2023) has the greatest scores on all metrics, averaging 84.8, 87.2 for rough-1, and 82.1 for rough-2. On the other hand, Muhammad Awais (2023) and Hassan Raza (2023) do worse, averaging 28.7 and 30.4, respectively. The suggested approach and Muhammad Aslam's approach show the best summarizing skills overall, pointing out areas that require more study and development in this area.

The effectiveness of the abstractive text summarizing method suggested in contrast to alternative strategies is summarized succinctly in Table 4.5. While comparable scores from other research offer background and illustrate the progress made in the field of Urdu text summary, the high ROUGE ratings attained by the suggested technique show its efficacy in producing high-quality summaries. Understanding the relative advantages and disadvantages of various summarizing strategies is made easier with the help of this table.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Using the mBART model with a multi-layer encoder-decoder architecture, the study's novel approach to abstractive text summarization for Urdu successfully tackles a number of particular difficulties presented by the language, including its intricate morphology, the extraction of semantic meanings, and dialectal variations that can highly impact summarization accuracy. As a result, the suggested method produces succinct, coherent, and well-structured summaries that accurately reflect the original content. Table 8 shows the performance of the model, which is quite encouraging. With an impressive BERTScore of 90%, the model demonstrated a high degree of semantic congruence between the reference texts and the summaries it produced. Furthermore, it achieved a BLEU score of 43%, indicating that the produced summaries and reference texts had a significant degree of lexical similarity. With ROUGE-1, ROUGE-2, and ROUGE-L scores of 80.3%, 74.3%, and 80.3%, respectively, the ROUGE measures further demonstrate the model's efficacy. These ratings show that the generated summaries preserve both surface-level information and underlying meaning by maintaining a fair mix of syntactic structure, lexical overlap, and long-range connections. The graphs in the findings section demonstrate how well-structured the produced summaries are, how contextually aware they are, and how comparable their lexicon is.

Despite these advancements, there are still issues to be resolved, especially concerning ensuring factual correctness, managing intricate sentence patterns, and the limitations imposed by the scarcity of high-quality Urdu datasets. This work is significant as it makes a substantial contribution to the field of low-resource language processing. By applying sophisticated summarization techniques to Urdu, a language with limited resources in the field of natural language processing, the method creates new opportunities for use in document summarization, news aggregation, and social media monitoring. These challenges, once addressed, could further increase the relevance and impact of this work in practical applications.

5.2 Future Work

Future studies should focus on growing the data set by including a wider variety of Urdu texts from different domains in order to improve the model's generalization and accuracy. It would also be helpful to investigate strategies for handling Urdu's intricate morphology and dialectal variances more effectively. Extended monitoring of model efficacy in several low-resource languages may provide additional confirmation of the suggested methodology. Furthermore, more thorough evaluation criteria that take semantic comprehension and context retention into consideration during Abstractive summarization should be the goal of future research.

Bibliography

[1] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, August 2018.

[2] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Pearson, 2016.

[3] J. A. Walsh, "Natural language processing in educational contexts: opportunities and potential pitfalls," Ph.D. dissertation, 2022.

[4] R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[5] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 3rd ed. Upper Saddle River, NJ, USA: Pearson, 2021.

[6] J. Huang and S. Liu, "The role of natural language processing in voice-activated virtual assistants," *Journal of Artificial Intelligence Research*, vol. 71, pp. 123–145, 2021.

[7] R. Binns, "Fairness in machine learning: Lessons from political philosophy," in *Proceedings of the 2018 Conference on Fairness, Accountability, and Transparency*. ACM, 2018, pp. 149–158.

[8] A. Nenkova and K. McKeown, "Automatic summarization," *Foundations and Trends® in Information Retrieval*, vol. 5, no. 2, pp. 103–233, 2011.

[9] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," *arXiv preprint arXiv:1908.08345*, 2019.

[10] R. Mitchell, F. Myles, and E. Marsden, *Second Language Learning Theories*, 3rd ed. Abingdon, U.K.: Routledge, 2019.

[11] F. Silvestri, "Mining query logs: Turning search usage data into knowledge," *Foundations and Trends® in Information Retrieval*, vol. 4, no. 1–2, pp. 1–174, 2009.

[12] N. Shafiq, I. Hamid, M. Asif, Q. Nawaz, H. Aljuaid, and H. Ali, "Abstractive text summarization of low-resourced languages using deep learning," *PeerJ Computer Science*, vol. 9, p. e1176, 2023.

[13] A. Nenkova and K. McKeown, "Text summarization: A review," *Foundations and Trends® in Information Retrieval*, vol. 5, no. 2, pp. 103–222, 2011.

[14] B. Mutlu, E. A. Sezer, and M. A. Akcayol, "Multi-document extractive text summarization: A comparative assessment on features," *Knowledge-Based Systems*, vol. 183, p. 104848, 2019.

[15] V. Dehru, P. K. Tiwari, G. Aggarwal, B. Joshi, and P. Kartik, "Text summarization techniques and applications," in *IOP Conference Series: Materials Science and Engineering*, vol. 1099, no. 1. IOP Publishing, March 2021, p. 012042.

[16] M. Lee, P. Liang, and Q. Yang, "Coauthor: Designing a human-ai collaborative writing dataset for exploring language model capabilities," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, April 2022, pp. 1–19.

[17] A. Nenkova and K. McKeown, "A survey of text summarization techniques," in *Mining Text Data*, S. S. Anand, Ed. New York, NY, USA: Springer, 2012, pp. 43–76.

[18] C. Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, 2004, pp. 74–81.

[19] I. Androutsopoulos and P. Malakasiotis, "A survey of paraphrasing and textual entailment methods," *Journal of Artificial Intelligence Research*, vol. 38, pp. 135–187, 2010.

[20] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2015, pp. 379–389.

[21] A. Maimon and R. Tsarfaty, "A novel computational and modeling foundation for automatic coherence assessment," *arXiv preprint arXiv:2310.00598*, 2023.

[22] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *arXiv preprint arXiv:1704.04368*, April 2017.

[23] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," *arXiv preprint arXiv:1908.08345*, August 2019.

[24] R. Nallapati, B. Zhou, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, February 2016.

[25] A. Nenkova and K. McKeown, "Automatic summarization," *Foundations and Trends® in Information Retrieval*, vol. 5, no. 2–3, pp. 103–233, 2011.

[26] S. Bhatia, "A comparative study of opinion summarization techniques," *IEEE Transactions on Computational Social Systems*, vol. 8, no. 1, pp. 110–117, 2020.

[27] H. Dang, K. Benharrak, F. Lehmann, and D. Buschek, "Beyond text generation: Supporting writers with continuous automatic text summaries," in *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, 2022, pp. 1–13.

[28] K. M. Chung, "Enhanced recommender systems by biclustering," Ph.D. dissertation, The University of Arizona, 2022.

[29] A. Nenkova and K. McKeown, "Automatic summarization," *Foundations and Trends® in Information Retrieval*, vol. 5, no. 2–3, pp. 103–233, 2011.

[30] H. M. Alghamdi and A. Selamat, *Computer and Information Sciences*. Publisher information if available, 2017.

[31] A. Elsaid, A. Mohammed, L. F. Ibrahim, and M. M. Sakre, "A comprehensive review of arabic text summarization," *IEEE Access*, vol. 10, pp. 38 012–38 030, 2022.

[32] F. Chollet, *Deep Learning with Python*. New York, NY, USA: Simon and Schuster, 2021.

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Regularization for deep learning*, 1st ed. Cambridge, MA, USA: MIT Press, 2016, pp. 216–261.

[34] C. Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, 2004, pp. 74–81.

[35] F. Kirstein, J. P. Wahle, B. Gipp, and T. Ruas, "Cads: A systematic literature review on the challenges of abstractive dialogue summarization," *arXiv preprint arXiv:2406.07494*, June 2024.

[36] R. Malkawi, M. Daradkeh, A. El-Hassan, and P. Petrov, "A semantic similarity-based identification method for implicit citation functions and sentiments information," *Information*, vol. 13, no. 11, p. 546, 2022.

[37] A. Nenkova and K. McKeown, "A survey of text summarization techniques," in *Mining Text Data*, 2012, pp. 43–76.

[38] M. Khan and M. Sadiq, "Abstractive text summarization in urdu: A comprehensive survey," *International Journal of Computer Applications*, vol. 175, no. 1, pp. 1–7, 2020.

[39] A. Raza, H. Raja, and U. Maratib, "Abstractive summary generation for the urdu language," *arXiv preprint arXiv:2305.16195*, 2023.

[40] A. Raza, M. Soomro, I. Shahzad, and S. Batool, "Abstractive text summarization for urdu language," *Journal of Computational Biomedical Informatics*, vol. 7, no. 2, 2024.

[41] M. Asif, S. Raza, J. Iqbal, N. Perwaiz, T. Faiz, and S. Khan, "Bidirectional encoder approach for abstractive text summarization of urdu language," in *Proceedings of the 2022 International Conference on Business Analytics for Technology and Security (ICBATS)*. IEEE, February 2022, pp. 1–8.

[42] N. Shafiq, I. Hamid, M. Asif, Q. Nawaz, H. Aljuaid, and H. Ali, "Abstractive text summarization of low-resourced languages using deep learning," *PeerJ Computer Science*, vol. 9, p. e1176, 2023.

[43] A. Faheem, F. Ullah, M. Ayub, and A. Karim, "Urdumasd: A multimodal abstractive summarization dataset for urdu," in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, May 2024, pp. 17 245–17 253.

[44] S. Iqbal and M. Rahman, "Challenges in maintaining semantic integrity in urdu text summarization," *Journal of Natural Language Processing*, vol. 32, no. 1, pp. 35–50, 2021.

[45] Y. Sunusi, N. Omar, and L. Zakaria, "Exploring abstractive text summarization: Methods, dataset, evaluation, and emerging challenges," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 7, p. [page numbers], 2024.

[46] N. Akram and R. Mehmood, "Improving readability and comprehension in urdu text summarization," *Journal of Computational Linguistics*, vol. 39, no. 3, pp. 567–585, 2021.

[47] S. Gehrmann, "Content selection for abstractive text summarization: A bottom-up approach," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[48] R. Katna, "Feature-based text summarization using latent semantic analysis," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

[49] C. Zhu, "Hierarchical network with cross-domain pretraining for text summarization," in *Proceedings of the International Conference on Natural Language Processing (ICNLP)*, 2023.

[50] J. Yao, "Joint summary generation for multilingual text summarization," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[51] A. Mohammed, "Statistical-based methods for arabic text summarization using machine learning," in *Proceedings of the Text Analysis Conference (TAC)*, 2023.

[52] A. Qaroush, "K-medoids clustering for arabic multi-document summarization," in *Proceedings of the International Conference on Arabic Language Processing*, 2023.

[53] A. Elsaid, "Transformer's encoder-decoder network for arabic text summarization," in *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2024.

[54] M. Aslam, "Sentence weight algorithm for enhanced information extraction in urdu text summarization," in *Proceedings of the International Conference on Text Mining and Natural Language Processing*, 2023.

[55] H. Raza, "Attention mechanisms in neural network architectures for abstractive text summarization," in *Proceedings of the International Conference on Natural Language Processing and Machine Learning*, 2023.

[56] M. Awais, "Transformer-based encoder-decoder model for abstractive summarization in urdu," in *Proceedings of the International Conference on Natural Language Processing and Artificial Intelligence*, 2023.

[57] R. Jain, "Biomedical text summarization using reward function based on word mover distance," in *Proceedings of the International Conference on Biomedical Informatics*, 2023.

[58] ——, "Biomedical text summarization using reward function based on word mover distance," in *Proceedings of the International Conference on Biomedical Informatics*, 2023.

[59] D. A. Khan, "Bio-semantic models for biological text summarization," in *Proceedings of the International Conference on Biomedical Text Mining and Bioinformatics*, 2022.

[60] R. Khan and H. Ali, "Advances in abstractive text summarization for low-resource languages: Case study of urdu," *Journal of Computational Linguistics*, vol. 29, no. 3, pp. 210–225, 2022.

[61] M. Awais and R. Nawab, "Abstractive text summarization for the urdu language: Data and methods," *IEEE Access*, 2024.

[62] M. Yarlagadda, H. R. Nadendla, and K. S. Rao, "Advancements and challenges in text summarization: An overview of methods and strategies in brief," in *Proceedings of SpringerLink*, 2023, online available: https://doi.org/10.1007/978-3-031-70001-9_8.

[63] M. Awais and R. Nawab, "Challenges in semantic representation for abstractive summarization of low-resource languages: A focus on urdu," in *Proceedings of the International Conference on Computational Linguistics and Language Resources*, 2023, pp. 112–125.

[64] M. M. Saiyyad and N. N. Patil, "Text summarization using deep learning techniques: A review," *Engineering Proceedings*, vol. 59, no. 1, p. 194, 2024.

[65] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in *Proceedings of the 2016 Conference on North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 93–98.

[66] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02, 2002, pp. 311–318.

[67] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *International Conference on Learning Representations*, 2020, presented at the conference, available at: https://openreview.net/forum?id=SkeHuCVFDr.

[68] M. Awais and R. M. A. Nawab, "Abstractive text summarization for the urdu language: Data and methods," *IEEE Access*, 2024.

[69] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 2002, pp. 311–318.

[70] K. Papineni, "Bleu: A method for automatic evaluation of mt," Computer Science RC22176 (W0109-022), Tech. Rep., 2001.

[71] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[72] K. Spärck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 60, no. 5, pp. 493–502, 2004.

[73] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," *arXiv preprint arXiv:1904.09675*, 2019.

[74] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017.

[75] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[76] A. Vaswani, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[77] J. M. List, "Automatic inference of sound correspondence patterns across multiple languages," *Computational Linguistics*, vol. 45, no. 1, pp. 137–161, 2019.

[78] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999.

[79] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. Sebastopol, CA: O'Reilly Media, Inc., 2009.

[80] N. Y. Habash, *Introduction to Arabic Natural Language Processing*. San Rafael, CA: Morgan & Claypool Publishers, 2010.

[81] Z. U. Rehman and I. S. Bajwa, "Lexicon-based sentiment analysis for urdu language," in *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*. IEEE, August 2016, pp. 497–501.

[82] G. Nunberg, T. Briscoe, and R. D. Huddleston, *Punctuation*. Cambridge, U.K.: Cambridge University Press, 2002.

[83] S. Zoya, S. Latif, R. Latif, H. Majeed, and N. S. M. Jamail, "Assessing urdu language processing tools via statistical and outlier detection methods on urdu tweets," *ACM Transactions on Asian Low-Resource Language Information Processing*, vol. 22, no. 10, pp. 1–31, 2023.

[84] C. D. Manning, *Introduction to Information Retrieval*. Cambridge, MA: MIT Press, 2008.

[85] R. Feldman and J. Sanger, *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data.* Cambridge, U.K.: Cambridge University Press, 2007.

[86] D. Jurafsky, *Speech and Language Processing.* Upper Saddle River, NJ: Prentice Hall, 2000.

[87] N. Schneider and N. A. Smith, "A corpus and model integrating multiword expressions and supersenses," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015, pp. 1537–1547.

[88] A. C. Stubbs, "A methodology for using professional knowledge in corpus annotation," Ph.D. dissertation, Brandeis University, 2013.

[89] J. J. Webster and C. Kit, "Tokenization as the initial phase in nlp," in *Proceedings of the 14th International Conference on Computational Linguistics (COLING 1992)*, vol. 4, 1992.

[90] G. Grefenstette, *Explorations in Automatic Thesaurus Discovery.* Springer Science & Business Media, 2012, vol. 278.

[91] W. A. Gale, K. W. Church, and D. Yarowsky, "A method for disambiguating word senses in a large corpus," *Computers and the Humanities*, vol. 26, pp. 415–439, 1992.

[92] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2021.

[93] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[94] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[95] A. Vaswani, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[96] K. Cho, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[97] A. Nenkova and K. McKeown, "Automatic summarization," *Foundations and Trends® in Information Retrieval*, vol. 5, no. 2–3, pp. 103–233, 2011.

[98] Z. Yang, B. Hu, A. Han, S. Huang, and Q. Ju, "Csp: Code-switching pre-training for neural machine translation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Nov 2020, pp. 2624–2636.

[99] R. Wang, "Efficient low-resource training with pre-trained deep neural networks," Ph.D. dissertation, Duke University, 2023.

[100] R. He, Z. Xing, W. Tan, and B. Yan, "Feature pyramid network for multi-task affective analysis," *arXiv preprint arXiv:2107.03670*, 2021.

[101] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[102] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[103] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Jul 2002, pp. 311–318.

[104] L. Chin-Yew, "Rouge: A package for automatic evaluation of summaries," in *Proceedings of the Workshop on Text Summarization Branches Out*, Barcelona, Spain, 2004.