National College of Ireland

Abstractive Summarization Using Neural Networks with Attention Mechanisms

MSc Research Project
MSc in Data Analytics

Kruthika Surendrakumar
Student ID: X22241965

School of Computing
National College of Ireland

Supervisor: Abdul Shahid

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Kruthika Surendrakumar
Student ID:	X22241965
Programme:	MSc in Data Analytics **Year:** 2024-2025
Module:	Research in Computing
Supervisor:	Abdul Shahid
Submission Due Date:	12/12/2024
Project Title:	Abstractive Summarization Using Neural Networks with Attention Mechanisms
Word Count:	8,153 **Page Count :** 23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Kruthika Surendrakumar
Date:	12/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Abstractive Summarization Using Neural Networks with Attention Mechanisms

Kruthika Surendrakumar
X22241965

Abstract

In the present world where a huge quantity of information is available, text summarization especially news articles text summarization has gained higher importance. Previous approaches to performing traditional extractive summarization appear to work well but fails in terms of providing precise and meaningful summaries. Previous methods and researched tended to suffer from low performance of generating coherent and informative summaries that would capture content of news articles. In contrast to the earlier works done in this area and to overcome the shortcomings, this study presents an improved abstractive summarization model for news articles using advanced sequence to sequence model framework. This study explores the methods of news text summarization with sequence-to-sequence models. Using the CNN/DailyMail dataset that offers more than 300,000 articles, the following piece of work focuses on the performance of basic and extended models to create 'compressed' and 'cohesive' summaries. The study evaluates two baseline models: LSTM and Bidirectional LSTM both Random Search regimen exclusion, and No-Attention paradigm. It then generalises this assessment to models with attention and learnt pretrained GloVe word vectors, namely Sequence to Sequence LSTM with attention and embedding, and BiLSTM with attention and embedding. The overall objective is therefore to evaluate the impact of these enhancements to the quality of summaries via ROUGE scores. It brings out an all round collection of data, cleaning and preprocessing of text data before proceeding to the final stages of model building. The performance is highly guarded and measured through the ROUGE metrics that allow the assessment of the quality of the generated summaries between the models.

Keywords: News text summarization, Sequence-to-sequence models, BiLSTM ,Attention Mechanism, Glove Embeddings, Abstractive Summarization.

1. Introduction

The process of taking large text and making it smaller retaining the important, useful details is called summarization.. Automating text summary has been of interest due to the potential to speed up the traditionally cumbersome task of generating a short summary of natural language processing (NLP) applications text. A few of those are text categorization, data retrieval, news summarizing, legal text summarization and headline generation (Gliwa et al. (2019)). Text summarization techniques are categorized under these two types: Abstractive and extractive summarization. Extractive summarizing involves picking out key lines or phrases directly from the original text using a scoring system; while, abstractive summarization reads through the content and paraphrasing it as hummanized summary (Zhang et al., (2020)).

The increase in information volume in our news feeds today poses a challenge to us readers filtering out the critical insights in each news article flooded in our news feeds in a timely and effective way. In order to meet this demand, businesses and content aggregators have started looking for automated solutions that can create short, informative summaries. Current techniques in summarization struggle to retain existing semantic depth of long articles, limiting the utility of created summaries. These shortcomings have given rise to research that is focused on augmenting sequence to sequence models by incorporating pretrained embeddings such as GloVe and additionally augmenting them using attention mechanisms. The results of these improvements are summarized with ROUGE scores. Specifically, the study analyzes whether the attention layers in combination with GloVe embeddings yield improved models for LSTM and BiLSTM models compared to baseline models lacking these features. The research then discusses the implications of these advanced models by focusing on a subset of the CNN/DailyMail dataset.

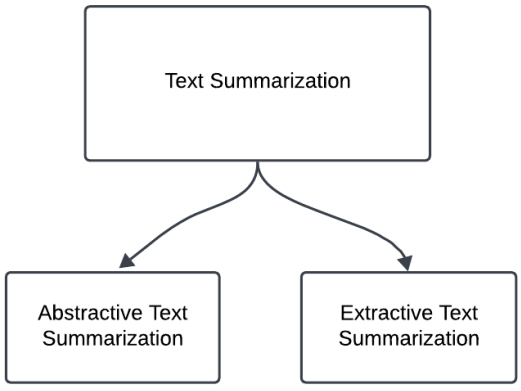

Figure 1: Types of Text Summarisation

2. Related Work

This article is the in-depth analysis of techniques used for automatic text summarization using several Deep Learning models. The review is divided into sub-sections as follows:

2.1. Extractive Summarization.

Text summary is broadly categorized into Two kinds of extractive and abstract approaches. In text extractive summarization relevant sentences are taken from the text, preserving the original content, in contrast abstractive summarization produces new sentences reformulating content from the source, more flexibly and fluently. Whilst being fluent, abstractive summarization struggles with content selection and coherence. To deal with these challenges, recent advancements integrate both techniques especially when it comes to summarizing long documents; Xiao and Carenini (2019) propose the use of a neural model to improve long document summarization by combining modeling of global and local context, and evaluated on the PubMed and arXiv dataset. Specifically, local context modeling is shown to be essential, and in fact even more important than capturing global context.

Liu et al. (2019) investigated the neural extractive models especially focusing on their architecture and learning strategies. Using methods such as BERT and reinforcement learning, they improved their performance on CNN/Daily Mail dataset with ROUGE scores of 42.69, 19.60 and 38.85. In addition, the large scale multi news dataset was introduced and the HiMAP model was used to tackle multi document summarization (MDS) by Fabbri et al. (2019). In turn, their hybrid framework combined single-document summarization (SDS) with extractive and abstractive methods and resulted in significant improvements. Extractive summarization was redefined by Zhong et al. (2020), as a similarity problem in semantic space. Enhanced performance on data sets like CNN/Daily Mail and WikiHow was demonstrated and further refinements are possible. Gupta et al. (2022) discuss transformer based models to summarize large text datasets, their differentiability between extractive and abstractive summarization methods and present them using a few examples. The study compared models trained on the BBC news dataset and compared their ability to produce natural and accurate summaries and looked at challenges of scalability and efficiency. Together, these studies emphasize the centrality of looking at text summarization as a combination of techniques (algorithmic and neural), architectures, and datasets to push the state of the art.

2.2 Abstract Summarization

Unlike extractive, which selects their own passages in the original text, abstractive summarization creates new sentences to represent what the text says. Currently, recent advancements concentrates in improving the content selection, coherence, and fluency. While

a Bottom Up attention mechanism was proposed by Gehrmann et al. (2018) to integrate a content selector to highlight key phrases. This method enhances efficiency, retraining ease, and performance yielding ROUGE-1, ROUGE-2, and ROUGE-L scores of 47.38, 31.23 and 41.81 on CNN-DM and NYT datasets.

For dialogue summarisation, Gliwa et al. (2019) introduced the SAMSum Corpus, a manually annotated dataset. The results indicate that summarizing dialogues are more challenging than doing so with news and show lower ROUGE scores on dialog tasks. All in all, this work highlights the necessity for creative models and evaluation metrics specifically designed for conversation data. Gupta & Gupta (2019) provide a survey of abstractive summarisation, tackling the generation of fluent summaries and evaluation. Exploring future directions, they gave a categorization comparing techniques, tools, and evaluation methods, and stressed the complexity of the task by the sheer inherent properties of language. Pegasus, a transformer based pre training method for abstractive summarisation, was first introduced by Zhang et al. (2020). It cuts out or muzzles the main sentences, then rebuild these sentences as continuos sequences. PEGASUS excels in 12 summarisation genres and sets the new state of the art on six benchmarks while also working well in low resource scenarios. Among seq2seq based neural abstractive models, Shi et al. (2021) classified the models based on saliency, fluency and readability. Techniques were categorized into network topology, parameter estimation, and decoding improvements. The authors benchmarked them on datasets like Newsroom and Bytecup and find that both models perform significantly better than an extractive model baseline and achieve strong ROUGE scores, demonstrating the model's applicability to different types of datasets. Together, these studies contribute towards abstractive summarisation by improving content selection, using pre-training objectives, and handling problems specific to summarising dialogue.

Study	Proposed Approachh	Challenges Faced	Results	Key Metrices
Xiao and Carenini, (2019)	Combined global and local context for long document summarization.	Benefits primarily from local context modeling.	Outperformed previous work on Pubmed and arXiv.	ROUGE-1, ROUGE-2, METEOR (specific scores not provided)
Liu et al., (2019)	Analyzed different model architectures, transferable knowledge, and learning schemas for extractive summarization.	Need to understand why neural models perform well and how to improve them.	Improved state-ofthe-art on CNN/DailyMail by a large margin.	ROUGE-1: 42.69, ROUGE-2: 19.60, ROUGE-L: 38.85

Fabbri et al., (2019)	Introduced MultiNews dataset and a combined extractiveabstractive summarization model.	Limited datasets for multidocument summarization.	Achieved competitive results on Multi-News dataset.	ROUGE-1: 43.47, ROUGE-2: 14.89, ROUGE-SU: 17.41
Zhang et al., (2020)	Proposed pretraining Transformers with a new selfsupervised objective; masked important sentences to generate summaries.	Need for effective pretraining objectives and systematic evaluation.	Achieved state-ofthe-art on 12 summarization tasks; performed well in lowresource settings.	ROUGE scores (specific scores not provided)
Shi et al., (2021)	Reviewed various seq2seq models for abstractive summarization and evaluated different neural network components.	Challenges in network structure, parameter inference, and decoding efficiency.	Benchmarked models on CNN/ Daily Mail, Newsroom, and Bytecup datasets.	ROUGE-1, ROUGE-2, ROUGE-L (specific scores not provided)
Gliwa et al., (2019)	Introduced SAMSum Corpus for abstractive dialogue summarization; compared performance with news corpora.	Dialogue summarization challenges and non-standard quality measures	Model-generated dialogue summaries achieved high ROUGE scores but lower human judgment compared to news.	ROUGE-1: 0.32, ROUGE-2: 0.30, ROUGE-L: 0.32 (correlation with human judgment)
Xiao and Carenini, (2019)	Combined global and local context for long document summarization.	Benefits primarily from local context modeling.	Outperformed previous work on Pubmed and arXiv.	ROUGE-1, ROUGE-2, METEOR (specific scores not provided)
Gupta and Gupta, (2019)	Reviewed recent papers on abstractive summarization; categorized methods and discussed challenges.	Complexities of natural language text; varying methods and tools.	Provided a thorough understanding of the field, listed methods, tools, and evaluation techniques.	No specific metrics provided

Xiao and Carenini, (2019)	Combined global and local context for long document summarization.	Benefits primarily from local context modeling.	Outperformed previous work on Pubmed and arXiv.	ROUGE-1, ROUGE-2, METEOR (specific scores not provided)
Gupta et al., (2022)	Compared transformer-based pre-trained models with various selfsupervised objectives.	Lack of tailored pre-training objectives for abstractive summarization; limited systematic evaluation	Achieved state-ofthe-art performance on 12 downstream datasets; excelled in low-resource settings.	ROUGE scores (specific scores not provided)

2.3 Pretrained Embeddings: GloVe in Seq2Seq Models

GloVe (GloVal Vectors for Word Representation) word vectors offer high dimensional, semantically rich representations and improve Seq2Seq models for the problem of text summarization. The pretrained embeddings are obtained from co occurrence statistics in large scale corpora, and outperform simpler representations, capturing structural word semantics. Other approaches using GloVe to refine the meaning of input data as well as enrich summaries semantically are described. DeepSumm (Joshi et al. (2023)) mounted topic models and word embeddings for single document summarization. RNNs with probabilistic topic distribution and an attention mechanism are used in DeepSumm to rank sentences on based of novelty, content, and position. It is tested on DUC 2002 and CNN/DailyMail datasets and reports ROUGE-1, ROUGE-2 and ROUGE-L scores of 53.2, 28.7 and 49, surpassing other existing approaches. The decoder qualitéé, to improve the performance of extractive summarization over the baseline, Solanki et al. 2024 applied a GloVe embedding with an LSTM based encoder-decoder model. To model the ontological relationships, embeddings were used which enhanced textual context understanding. The model was evaluated on the Kaggle news dataset and achieved BLEU score of 59.4% and cosine similarity of 50.2%, which is substantial increase in model quality. First, these works show that incorporating features from GloVe embeddings can help improve both extractive as well as abstractive summarization by obtaining better semantic and contextual representation.

2.4 Attention Mechanisms in Seq2Seq Models

It was found that attention mechanisms in Seq2Seq models are essential for improving abstractive text summarization. In Kumar and Solanki (2023), the authors proposed a Transformer based model with a self-attention mechanism, denoted T2SAM. T2SAM outperformed baseline models by addressing coreference identification as well as improving textual cohesion. It was trained on multiple datasets and decreased training loss from 30.58 to 1.82 in 30 epochs, achieving 48.5% F1-Score accuracy indicating substantial improvements in summary quality and coherence. A Seq2Seq model with attention architecture has been

developed by Sultana et al. (2022), for Bengali news summarization by applying Bidirectional RNN and LSTM. Further, with preprocessing and word embedding on a dataset of 30 Bengali newspapers, model parameters such as training loss reduced to 0.001 and accuracy improved from 44.48% to 56.19%. This work successfully addressed the problems of Bengali language summarization, producing summaries that are more succinct and better than baseline models. Pan et al. (2019) understood how implicit and contextual informations are handled in text summarization. In order to incorporate scene and topic context into Seq2Seq models, they introduced an external attention mechanism using Latent Dirichlet Allocation (LDA) based word topic distribution. This approach is tested on CNN and Daily Mail datasets and outperforms baselines as a strongly robust approach to uncover hidden context and advance sentence summarization. In addition to these, the versatility of attention mechanisms to improve contextual understanding and ensure that the summary is accurate was shown.

3. Research Methodology

In this section of the report we will explore the approach taken for conducting the study that is being presented.

3.1 Dataset Description

Among the most popular datasets for text summarizing model training and testing in particular for extractive and abstractive summarization tasks are CNN/DailyMail. Over 300K news items are included from CNN and the Daily Mail, plus brief, human written summary highlights that serve as synopses of the articles. These features make the dataset very useful for training algorithms that generate summaries. The dataset, initially used for abstractive question answering and machine reading comprehension, is challenging because it requires summarization and contextual information.

The story's content, associated highlights and a unique integer id (SHA1 hash of the story's url) are in every dataset item.Although widely used, the database possesses problems such as the incorrect factual correctness, coherence, and fluency; and suited adjectives in the summaries.an 300,000 news items from CNN and the Daily Mail are included, along with succinct, human-written highlights that act as synopses of the pieces. Because of these features, the dataset is very useful for training algorithms that create summaries. The dataset, which was initially created for abstractive question answering and machine reading comprehension, is difficult since it calls for both summarization and contextual information.

Every dataset item contains the story's content, associated highlights, and a unique identifier (SHA1 hash of the story's URL). Despite its extensive use, the dataset presents difficulties by including problems with factual correctness, coherence, fluency and the suitability of the

summaries adjectives. When preparing the data to train the model, besides cleaning the text, reducing interference and creating token and word embedding matrices of them are required. The word vectors used within GloVe, and the other pretrained word embeddings, leverages word vectors to add to the semantic depth of the summaries. The CNN/DailyMail dataset, when used alongside the more complex methods such as attention mechanisms and deep learning models, provides a wealth of help for developing text summarization algorithms at large.

3.2 Data Preparation and Setup

The environment and data preparation is done before model implementation. During model training and evaluation the TensorFlow logging API is used to control what is output so that only the proper information is displayed. To accomplish this, we use a command: logger = tf.get_logger() to create a special logger that is directed to take care of TensorFlow activities. K.clear_session() is called to clear the session if there are conflicts from previous models or layers to prevent memory leaks so that it's safe to reset the model for another comparison.Also, irrelevant warning messages are filtered out with Python's warnings.filterwarnings('ignore') command so things important can be focused on or the data outputs. A couple of text preprocessing techniques are applied in the preprocessing stage: stopwords removal (using the Natural Language Toolkit (NLTK)). Commonly, noise in data is reduced by removing stopwords such as "and," "the," or "is." It is then made sure the stopword list is updated using nltk.download('stopwords') so that the dataset can be cleaned before processing further. Those are the steps which help to design a clean and productive framework, which tests and trains the text summarization models in the best way.

3.3 Importing Libraries

Libraries that can handle streaming data, model and model evaluation, have been imported to text summarization. Most basic operations on files use OS and re for regular expressions; reading and writing CSV use csv. It provides the tools for natural language processing: nltk.tokenize.word_tokenize,—for tokens, nltk.tag.pos_tag,—for parts of speech, nltk.WordNetLemmatizer,—for lemmatization and nltk.corpus.stopwords,—for stopwords removal. Models built using layers such as LSTM, Bidirectional, Embedding, and Dense are all built for deep learning using Keras and TensorFlow. Plots of the data distributions and of the model performance metrics are done using Matplotlib and Seaborn visualization tools. The frequent words are visualized using WordCloud from the wordcloud package and NLTKs STOPWORDS. Pickle and load_model from TensorFlow makes saving and loading models for model persistence easy. Moreover, text tokenization and padding are handled by TensorFlow's tokenize, and pad_sequences, respectively. Early Stopping along with ReduceLROnPlateau techniques prevent overfitting reduce the learning rate. To evaluate

model, rouge_scorer from rouge_score package has been used and train_test_split has been used for performance analysis from sklearn.model_selection. It offers comprehensive set of libraries for effective training of the model and the evaluation of model.

3.4 Data Loading and Initial Processing

The first process of preprocessing data for the task of abstractive text summarization on the CNN/DailyMail dataset is data loading using the pandas library. The dataset is read from CSV file placed at train. csv data by using the function pd. read_csv(csv file, numberOfRowsInSection: 11000 for efficiency's sake. Evaluating the models and training the models with minimal burden to computational resources can be done with subsets of this. Since the only content data that matter to us is the textual content, we can simply drop all unnecessary columns during the data processing. For example id column containing SHA1 hash of the articles' URL doesn't help us in delimiting the given articles for summarization and can be removed 21 using the drop() function with axis="columns" just like below. The resulting DataFrame consists of two critical columns: Two distinct signs of the new cultural identity are noted and emphasized in detail. The high point of this project is the 27 summaries written by the authors of the original articles and the full text of the news articles that are embedded with the CA News. Input text and desired output text are the summarization models that train on these columns. Because data loading and preprocessing was carried out in such a structured manner, every data set should be in the right format in order for it to pass through the text cleaning processes, tokenization and the data be ready for training of the models. Therefore filtering from the pointless columns makes a path to profound and sharp division of authentic profitable information for recognizing just as scale summary so it increases the chances of the model.

3.5 Text Cleaning

Preparation of data for abstractive text summarization requires a very crucial step of text cleaning. class Util: The Python function clean_text() allows to perform effective text preprocessing using NLTK and regex. First of all, it converts the input text to lowercase so as to normalize differently capitalized text. Then, the text is tokenized and contractions are expanded with a pre defined dictionary. But regex functions can be used to remove unwanted elements such as URLs, HTML tags, punctuation and special symbols. That means that among other things, it strips away the non-impactful characters to make a cleaner version of the text. The function also deals with apostrophes and other punctuation marks which could break up words, replacing them by a space. If remove_stopwords True, remove stopwords (example; articles, conjunctions), an optional step for the model to concentrate on more relevant content. These are stored in separate lists: clean_texts and clean_summaries. This guarantees that the articles and its highlights both underwent preprocessing to be used in later

tokenization and model training. This process greatly cleanses the data the good results of text summarization models.

3.6 Data Preparation for Model Training

As the first step, texts need to be cleaned and then formated to fit text summarization models. A new DataFrame called dataframe1 is created with the cleaned articles under the 'text' column and their corresponding summaries under the 'summary' column. NaN values denote incomplete summaries, which are replaced with the mean value (representative to all articles) before using the dropna() function in order to remove any null values from the dataset, leaving every article with a valid summary. Special tokens (e.g. <start> and <end>) are added to each summary to preserve the structure of summaries for sequence to sequence models. By including these tokens, the model is able to figure out the start and end of each of the summaries, which is central for creating accurate text. This is done using the apply function with a little modification on each entry in the "summary" column. After cleaning, the DataFrame contains 11,000 rows of an article along with its summary with start and end tokens. Finally structured data pipelined to tokenization and model training that can be given as the well defined input for text summarization models. To be able to generate coherent and accurate summaries the data needs to be carefully prepared.

3.7 Data Visualization

Figure 2 is the histogram of text lengths in the dataset, where horizontal axis is the number of characters in each text from 0 to 14,000, and the vertical axis is the frequency of texts in each character range.e. The distribution is positively skewed and peaks at around 2,000—a couple 2,500 characters—meaning that lots of the texts in the dataset are that long or longer. For shorter texts the frequency rises very steeply, and then decreases slowly for longer texts, implying that texts below 1000 characters and beyond 6000 characters are quite rare.

The majority (97%) of the texts are between 1000–4000 characters; the shortest 3% of texts fall within this range. A very small number of longer texts is responsible for the rightward extension of the curve — as text length increases, the frequency of these texts decreases. This histogram provides useful info about the text size is in the dataset which is very helpful for things like natural language treatment and substance examination since the size of the content can affect the execution of the model and the strategies utilized for handling it.

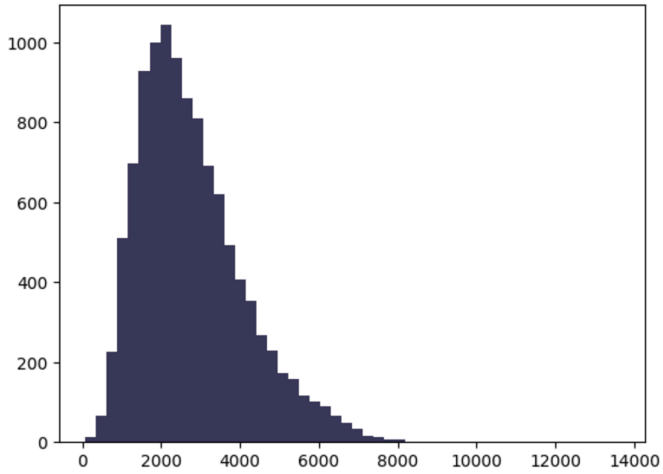

Figure 2: Histogram of Distribution of Text Lengths

As shown in Figure 3, the summary length histogram of the dataset is in the form of a histogram of the summary lengths (0–1000 characters) on the x axis and the frequency of the summary lengths at each certain length on the y axis. Moreover this distribution is positively skewed and there is a sharp peak around 250-300 characters which implies that most of the summaries in dataset are of this length or near this length. A second peak is visible around 350-400 characters, which again shows a commonly held summary length. Only 4% of summaries get longer than 600 characters, we are seeing pretty rare summaries longer than 500 characters. About this distribution, it concentrates on shorter summaries, outlining the usual length and verbosity of summaries that can be used for checking whether an automated summarization model is sufficiently efficient and describes it well.

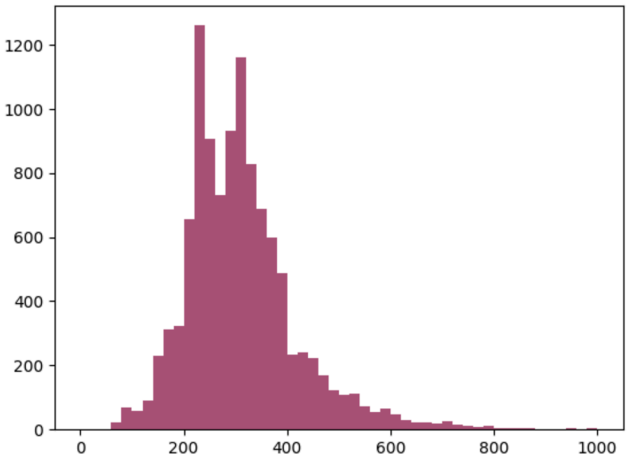

Figure 3 : Histogram of Distribution of summary lengths

4. Design Specification

The last stage of data processing converts the data to the form friendly to deep learning models, in particular, the tokenization, one hot encoding, padding the sequences and preparing pre trained vector matrix of the tokenized data. First approach: Initially, the articles text is tokenized using Keras's Tokenizer class which maps each word with a distinct integer

index according to their count in the dataset. This data is set up for on hot encoding, setting up the data to a MAXIMUM of 110,788 unique tokens. This converts the tokenized words to numerical sequences of integers, and then 1-hot encoded vectors so the data is ready to be given to a model. Secondly, padding is applied to make the sequences of same lengths, critical to a deep learning model taking batches. Keras pad_sequences function pads sequences shorter than 800 tokens by zeros at the end. And this means training, validation and testing data sets stay consistent. The training data has 110,788 unique tokens, the number of which increases to 246,031 after padding.

The summary data faces a smaller vocabulary size of 36,688 tokens as the summaries are shorter and less diverse; thus, a similar tokenization and encoding process is used. Summaries are padded to a constant length of 150 tokens. A semantic similarity between words is captured by setting embedding dimensions equal to 300, so that they match the pre trained GloVe embeddings. Subsequently, tokenization, one hot encoding, and padding is done on the entire dataset (training, validation and testing) to make the data ready to be used in building and training sequence to sequence models with attention mechanism, necessary for abstractive text summarization.

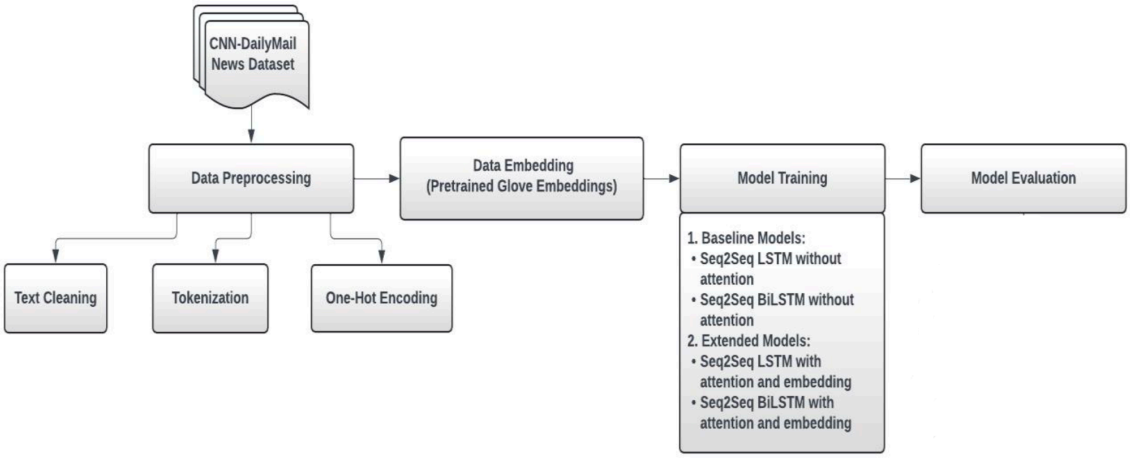

Figure 4: Architecture of proposed workflow

5. Results And Discussion

This section of the report discusses the implementation of the models and presents the results obtained for the experiments conducted. The implementation of the system is done using the Python development language using Google Colab. Google Colab is used for its access to the powerful CUDA GPUs. The dataset used in the study is first uploaded on the Google Drive. The dataset then can be accessed by mounting the Google Drive in the Colab environment. We have subscribed to Google Colab pro account and used High Ram A100 as our hardware accelerator to execute our model faster. Keras is the primary library used for the implementation of the deep learning models.

5.1 Case Study 1:

Sequence-to-Sequence LSTM model: It is an encoder and a decoder, Sequence to Sequence LSTM without embedding model, with the aim of performing abstractive text summarization. The model consists of two main components: The most familiar of the two models were the encoder and decoder.

Encoder: The encoder begins with an input layer that takes as input sequences of text with a length less than 800 tokens at most. The embedding layer, which is non trainable, assigns a 300-dimensional vector to each token, using a matrix of pre trained embeddings. This layer into convert the input sequences into dense vectors. The input to the embedding layer is passed through LSTM layer with the latency of 128. This LSTM layer takes the sequences and gives both the sequence of outputs and the last hidden and cell states which are very important when initializing the decoder.

Decoder: The decoder is built to include an input layer that can take as input sequences of tokens of any length to help start decoding. As in case of the encoder the decoder also consists of an embedding layer which is pre trained to embed tokens to dense vectors. Then these are fed into LSTM layer that has latent dimension 128 and returns sequences of outputs and updated hidden and cell states. This LSTM layer uses Dropout and a recurrent Dropout to help reduce the overfitting problem. Then output of the decoder LSTM passes through TimeDistributed Dense layer that does the dense transformation on each time step to get the probability distribution of vocabulary for each token. In cases of a single framework, encoder and decoder are incorporated into the whole structure developed using the Keras Model API.

Model summary: 49,200,348 parameters total, of that number 5,107,548 parameters are trainable and 44,092,800 parameters are not trainable. Basically nt parameters are the pre trained embeddings used used in case of training encoder and decoder. The proposed approach is sequence to sequence learning for summarization and use LSTMs to capture temporal information from the input text.

Metric	Training	Validation
Loss	5.6213	2.7141
Accuracy	0.6850	0.6818

Table 5 : LSTM without embedding

5.1.1 Model Training and Model Inference

For abstractive text summarization, the Sequence to Sequence LSTM model without embedding was used for implementing the model. A sparse categorical cross entropy at a loss with an RMSprop optimizer and accuracy as the metric as trained. For execution of this training, 30 finite epochs were used and a batch size of 32, with training and validation datasets used to measure the performance. x_train_padded and y_train_padded had a preceding token, shifted from the summaries, to enable learning. From the previous strings of words the model learned to generate the next word in a sequence and synthesized summaries from encodings of the input strings. A different model enc_model has also been developed for the inference, for getting encoder hidden and cell state required to initialize decoder.

For sequence generation, the states were designed to be utilized by the decoder model (dec_model). First, it was incorporated with start token and then it predicted the next tokens sequentially until either it predicted the encoding of stop token or prediction of the maximum number of tokens was reached. Test article summaries were produced over this process. The original summaries in the collection were then matched with the auto generated summaries. The full articles, the summary produced by the tool on each article, and the summary generated by the present model were then written to a CSV file for further analysis with the outcome.

Sample Inputs	Precision	Recall	F-Measure
0	0.090909	0.075472	0.082474
1	0.136364	0.093750	0.111111
2	0.045455	0.058824	0.051282
3	0.045455	0.046512	0.045977
4	0.000000	0.000000	0.000000
5	0.113636	0.096154	0.104167
6	0.113636	0.128205	0.120482
7	0.022727	0.020833	0.021739
8	0.045455	0.044444	0.044944
9	0.159091	0.088608	0.113821

Table 6 : Rough Score

5.2 Case Study 2: Sequence-to-Sequence BiLSTM model

There is a strong and reliable Basic Bidirectional Long Short Term Memory (BiLSTM) Encoder-Decoder model that is appropriate for sequence to sequence problems and works with no pre-trained embedding. The encoder starts with taking an input layer where sequences of length max_text_len are taken from the following: These sequences are fed into an embedding layer which has each word embedded into a vector of size 'embed_dim' and this layer is also trainable. Subsequently, there is a BiLSTM layer having the capacity to process the received word embeddings in the forward and backward mode, so that it gives out the outputs as well as states. Bidirectional processing enhances the identification of context from the left and right of the sequence to the model. Just like the case of BiLSTM; the output is summarized to form enc_h and the decoder_h which serve as the first state for the decoder. Decoder also has the word embedding layer which maps the tokens in input to the high dimensional vector, which is also optimized with rest of the parameters.

In the next step, a conventional LSTM layer processes these embedding, while using 'fused' hidden and cell states of the encoder as 'self' initial states. This LSTM layer generates sequences of the output vectors which are then passed through a TimeDistributed Dense layer with softmax activation function for making the next word of sequence prediction at each time step. The model is compiled to accept sparse categorical cross entropy loss with rmsprop optimizer to explain the ability of the model in predicting a sequence. This model includes 54,402,780 total parameters – all of which are trainable – and affords you end-to-end learning from your training data, making it suitable for tasks where you'll need to capture subtleties of contextual information and apply sequential processing.

Metric	Training	Validation
Loss	4.3357	2.6389
Accuracy	0.7002	0.6946

Table 7: BiLSTM without embedding

5.2.1 Model Training and Model Inference

A Sequence to sequence Bidirectional LSTM without embedding was developed and trained using sparse categorical cross entropy as the loss function, RMSprop optimizer and sought for accuracy. In this case the model was fit on the paddle sequences 36 x_train_padded and y_train_padded where y_train has its last token removed and reshaped to input and output format. The training step was as follows: Training was carried out using 30 epochs with a batch size of 32 while validation was performed using the same epoch of the validation data.

The encoder was configured to be used with model inference to generate hidden states (enc_h) and cell states (enc_c). The predictions are obtained by the decoder, which was initiated by the encoder's states. generate_summary was the fluid EH AR sequence generation function which initially forecasted the initial states from the encoder and thereafter purely forecasted the tokens using current token and current states in a loop. Generation loop continued till an EOS token were generated or till the limit of maximum number of tokens were generated.

Sample Inputs	Precision	Recall	F-Measure
0	0.074074	0.075472	0.074766
1	0.166667	0.140625	0.152542
2	0.055556	0.088235	0.068182
3	0.055556	0.069767	0.061856
4	0.018519	0.016949	0.017699
5	0.092593	0.096154	0.094340
6	0.092593	0.128205	0.107527
7	0.018519	0.020833	0.019608
8	0.055556	0.066667	0.060606
9	0.148148	0.101266	0.120301

Table 8: Rough Score

5.3 Case Study 3: Sequence-to-Sequence LSTM model with attention and embedding

With pre-trained word vectors of GloVe embeddings, the improved Sequence-to-Sequence LSTM with attention performs the best. First, word embeddings are pre trained and are imported and stored in vectors t_embed and s_embed that are the source and target vocabulary. Then, the Embedding layers for the encoder and decoder are used by these embeddings as shown in Fig 3. Here an LSTM layer is used with L2 regularisation of 0.001 and output size of 128 latent dimensions per time step which will give sequences of hidden states and cell states to encode in input sequences. In the other hand, the encoder is an LSTM layer with similar dimensions as the one in the decoder that emits outputs from the salutatory states from the encoder. A specific class, 37 AttentionLayer, is needed to implement the attention techniques, computing a matching between the encoder's outputs and decoder's states, and obtaining context vectors that point to useful portions of the input sequence. The attention mechanism of this model pays attention to a large part of the inputs when generating every word of the sequence of the output words.

The output of this decoder and this attention layer is passed to the next layer, which is TimeDistributed dense layer with ReLU activation at each step of the decoding sequence (as the decoder is giving us one output at each step) then we have softmax activation function which gives another output, a probability for each word in the dictionary at each step during decoding. The model has in total 53,865,308 parameters spread over input embedding layer that has 70,000 parameters and 53,795,308 parameters in the output MLP layers; with 9,772,508 of these trainable, and the rest froze and pre-trained indices assumed embeddings that won't be learnt during the training. The model plot file presents the visualisation of this architecture by showing the layers and how they are connected and the shapes of the layers. Integrating attention enables the model to make use of long range dependencies, be more focused at specific parts of input sequences, and might help improve sequence generation in these downstream applications of, for example, text summery or translation.

Metric	Training	Validation
Loss	4.6510	2.6754
Accuracy	0.6859	0.6820

Table 9: LSTM with attention and embedding

5.3.1 Model Training and Model Inference

Seq2Seq includes some attention and pre trained embedding and trained by loss of sparse_categorical_crossentropy with optimizer of rmsprop, with main criterion of accuracy. Training the model works well with the fact that the input sequences are padded and the targets are reshaped (reshaped) correctly to the output dimensions. After this a model can be trained for several iterations with epochs to equal 5 and batch size to equal to 32 after which the model is evaluated on the validation data. The hidden states and contextvector, produced from the input sequence, are computed in the encoder model for infererence. It takes the states from the encoder it is initialized with, and generates output sequences. An attention layer is used to calculate context vectors to guide the decoding process according to information from the input sequence and to concatenate these vectors to the decoder outputs. Finally these concatenated vectors are used to make the last layer dense layer that becomes our final output. To acquire the summaries, the model keeps spitting out the next token until hits the eos token or whatever the chosen max 38 length. By computing summary it stores for the later comparison with the original so test inputs can be evaluated.

Sample Inputs	Precision	Recall	F-Measure
0	0.049505	0.094340	0.064935
1	0.069307	0.109375	0.084848
2	0.029703	0.088235	0.044444
3	0.019802	0.046512	0.027778
4	0.029703	0.050847	0.037500
5	0.049505	0.096154	0.065359
6	0.049505	0.128205	0.071429
7	0.009901	0.020833	0.013423
8	0.029703	0.066667	0.041096
9	0.089109	0.113924	0.100000

Table 10: Rough Score

5.4 Case Study 4: Sequence-to-Sequence BiLSTM model with attention and embedding

For this work, Bidirectional LSTM, attention mechanism, and the GloVe pretrained embeddings were adopted to handle some complicated text summarization tasks in a Seq2Seq model. The model uses word embedding from GloVe which uses only word that has 300 dimensional vectors. First these embeddings are read from a file, they are converted to a dictionary and used to initialize matrices that are used for embedding the encoder and the decoder. In particular, t_embed and s_embed are matrices obtained from GloVe embeddings of respective source and target vocabulary, which in this case refers to word embeddings of the considered layers of the model that are pre trained on syntactic aspects. In particular, the encoder consists of an input layer and an embedding layer which is initialized with t_embed. A Bidirectional LSTM Layer is used which is contextual layer, it does take forward as well as backward information of the sequences. The two hidden states from either direction are stacked into one vector of context, enc_h and enc_c (from encoder). The same model is used to insert input into the decoder and it is then fed to LSTM layer with dropout to prevent overfitting. The attention mechanism, realised through a user-defined AttentionLayer, calculates attention scores: to produce the product of the prior, from the encoder, and the current from the decoder. It's done with trainable weight matrices that compute alignment scores and hence context vectors. Decoder's LSTM outputs are connected to the context 39 vectors, and the resulted vectors are final inputs for the dense layer.

In the end, before returning the sequence of the final hidden state to really predict some more on the vocabulary, we have the final layer of our model which is a TimeDistributed Dense layer. However, the learning and the generation of the summaries is proper with the mechanism of attention and the GloVe embeddings; the model parameters; total parameters 63,798,236 (include trainable parameters 30,561,836).

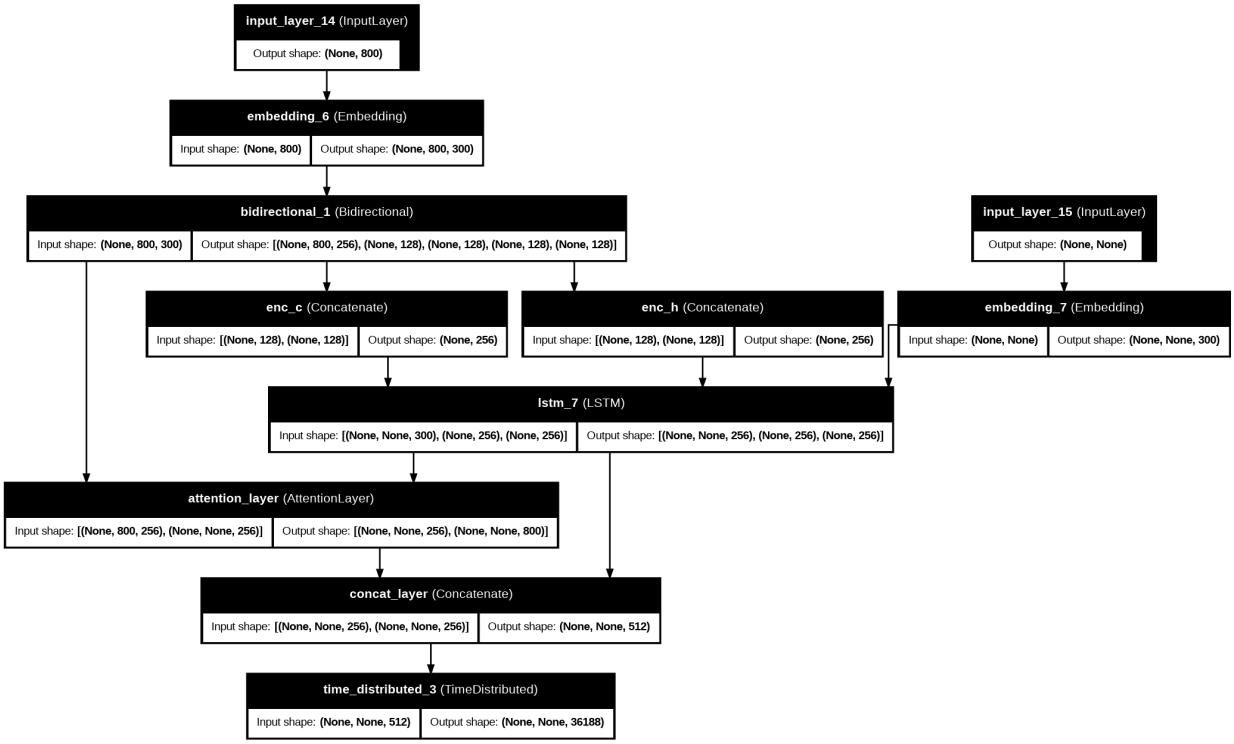

Figure 11: Sequence-to-Sequence BILSTM model with attention and Embedding

Metric	Training	Validation
Loss	3.7042	2.6380
Accuracy	0.8277	0.7029

Table 12: BiLSTM with attention and embedding

5.4.1 Model Training and Model Inference

This is the complete model used to train; the Sequence to Sequence BiLSTM model with attention and embedding, Adam optimizer and sparse categorical cross entropy was used to train the model. Sequence prediction was used in training the model and inputs and outputs, embedding dimension set at 300 and GloVe pre trained word vectors were applied. These will

output our encoder sequences and hidden states in our encoder model which we used BiLSTM layers; while our decoder was the LSTM using attentions in 40 (hopefully this will increase context relevancy). Encoder and decoder models were trained with a batch size of 32 up to a fixed number of epochs and the encoder model that had been trained had subsequently been saved for inference. In case of model inference, the generate_summary function is used where the input sequences are encoded which results in the fixed length vector, and decoder then generates the summary. The decoder keeps predicting tokens in sequence from startof-sequence token till end-of-sequence token is reached or maximum tokens is achieved.

Sample Inputs	Precision	Recall	F-Measure
0	0.1875	0.056604	0.086957
1	0.2500	0.062500	0.100000
2	0.1875	0.088235	0.120000
3	0.1875	0.069767	0.101695
4	0.0625	0.016949	0.026667
5	0.2500	0.076923	0.117647
6	0.2500	0.102564	0.145455
7	0.1250	0.041667	0.062500
8	0.1875	0.066667	0.098361
9	0.3750	0.075949	0.126316

Table 13: Rough Score

And then, by appling pre-trained GloVe embeddings with 300 dimensions in each of the words in the input sequence that are dense (i.e semantic meaningful) representations for the words. The alignment score, which is computed by the attention layer between the encoder and the decoder states enhances the model to give different weights to different parts of the input sequence and thus can summarize a text more accurately. By introducing attention, embedding and improvement to the model, the Sequence-to-Sequence BiLSTM model has demonstrated some practical benefits in summary creations due to the model's advanced structure able to find more detailed information and produces the smooth text that makes sense given the context.

	Article	Original Summary	Model Output
0	emily davies published 05 42 est 19 february 2...	<sostok> new bill would allow permit holder to...	the dog is the rest of the year s foundation ...
1	islamic state fanatics claim constructed dirty...	<sostok> chemical was stolen from mosul univer...	isis militants were seized by a rebel forces ...
2	associated press daily mail reporter published...	<sostok> tyler seddon has always wanted to be ...	the former police officers were held at the u...
3	huge difference wages premier league football ...	<sostok> premier league managers earn an avera...	chelsea are bottom of the premier league cup ...
4	antioch california cnn investigators completed...	<sostok> police say they have not eliminated p...	police remove the body of the body of the bod...
5	cnn united states donate 15 million auschwitz ...	<sostok> new u s poland amend missile agree...	americans are calling for the u s to visit th...
6	iraqi man shot killed seven isis militants act...	<sostok> iraqi man opened fire on isis militan...	mohammad abdul ali al shammari was released f...
7	daily mail reporter 23 year old arizona woman ...	<sostok> anna areola hernandez of glendale a...	police reportedly offered a girl to a girl wi...
8	mourner got drunk friend funeral woke hundreds...	<sostok> james o kane 22 thought he was in o...	flight was forced to return to the airport in...
9	anthony bond published 11 40 est 9 may 2013 up...	<sostok> victim fell prey to phishing scam a...	police say thieves could have fallen true num...

Figure 14: Comparison of Article Summaries - Original vs. Model Output

6. Conclusion and Future Work

This paper provides a novel robust approach based on the sequence-to-sequence models combining the BiLSTM with attending mechanisms and the Glove embeddings to generate abstractive news summarization in this study. In addition, our strategy tries to minimize the shortcomings pointed out in earlier work, such as varying quality summaries and inability to handle contextual information issues. These bidirectional LSTM units and attention processes accurately learn the relationship between words and phrases and therefore supply a richer contextual summary. The second benefit of the selected model is based on using the pretrained Glove embeddings to expand the semantic dimension of the incoming data.

Finally, the results of the evaluation that have been computed using ROUGE analyses show that the proposed Fuzzy-Based text summarization method is both more precise, more recall and less erroneous than traditional text summarization methods. This confirms that this proposed method of embedding BiLSTM, attention mechanism and pretrained embeddings together produce better quality summaries, which are closer to the source content. These are our major conclusions and contributions: The studies showed that the potential of improving text summarization task can be achieved by enhancing development of the more and more sophisticated neural network structures. The findings also reflect how using advanced approaches and effective resources for model improvement favorably boosts the results, and thus becomes a guideline for future research in this field.

6.1 Future Works

There are several roads in terms of future research and implementation of text summarization models especially for the news articles. Further studies could be conducted on the continued improvement of model performance such as by extending the scope of the application of the suggested model or solving the remaining problems. To begin with the new, more sophisticated word embeddings customizing text context might help to makes the quality of the summary better. Better semantic properties of transformer-based embeddings (eg: BERT, GPT) enable more research to be done with recent advances in transformer based embeddings.

Furthermore, it may be interesting to see 44 how these embeddings can be made as potential additions or replacements for existing 36 systems and frameworks, and whether hybrid systems (adding these embeddings to the existing methods as a whole) may be more fine grained and/or more coherent in other situations. The second direction of future research deals with the improvement of applied attention structures of the summary generation

problem models. The current experiment uses basic attention procedures, however, using more complex forms of attention like multihead or selfattention may help the model try focussing to different parts of the text at one time, which can yield better quality of summaries. It has also been expanded to include further data in further news sources and further domains, which would contribute to the creation of more general models. At present, the dataset that the work is based on is one ; however, it will be meaningful to include data from other news sources and genres for the model to apply well to the other contexts and topics.

Additionally, as the model is fine tuned to carry out shorter summarization tasks similar to the summarization of the news article where the focus topic or summarizing of key events is stated, the model becomes finer to the user's needs. Further directions include creating real time summarization options, utilizing incremental learning more, and producing convenient interfaces for live summarizing. Lastly, the models will be compared to other higher standards, and participation of real users in the feedback for improving upon the model will help and ensure that the summarization tool would remain effective, even if we see shifting in the information landscape.

References

Zhong, M., Liu, P., Chen, Y., Wang, D., Qiu, X. & Huang, X. (2020) ‘Extractive summarization as text matching’, *arXiv preprint arXiv:2004.08795*.

Xiao, W. & Carenini, G. (2019) ‘Extractive summarization of long documents by combining global and local context’, *arXiv preprint arXiv:1909.08089*.

Gupta, A., Chugh, D., Anjum, & Katarya, R. (2022) ‘Automated news summarization using transformers’, in *Sustainable Advanced Computing: Select Proceedings of ICSAC 2021*. Singapore: Springer Singapore, pp. 249–259.

Zhong, M., Liu, P., Wang, D., Qiu, X. & Huang, X. (2019) ‘Searching for effective neural extractive summarization: What works and what's next’, *arXiv preprint arXiv:1907.03491*.

Fabbri, A.R., Li, I., She, T., Li, S. & Radev, D.R. (2019) ‘Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model’, *arXiv preprint arXiv:1906.01749*.

Shi, T., Keneshloo, Y., Ramakrishnan, N. & Reddy, C.K. (2021) ‘Neural abstractive text summarization with sequence-to-sequence models’, *ACM Transactions on Data Science*, 2(1), pp. 1–37.

Gliwa, B., Mochol, I., Biesek, M. & Wawer, A. (2019) ‘SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization’, *arXiv preprint arXiv:1911.12237*.

Gupta, S. & Gupta, S.K. (2019) ‘Abstractive summarization: An overview of the state of the art’, *Expert Systems with Applications*, 121, pp. 49–65.

Zhang, J., Zhao, Y., Saleh, M. & Liu, P. (2020) ‘Pegasus: Pre-training with extracted gap-sentences for abstractive summarization’, in *International Conference on Machine Learning*. PMLR, pp. 11328–11339.

Gehrmann, S., Deng, Y. & Rush, A.M. (2018) ‘Bottom-up abstractive summarization’, *arXiv preprint arXiv:1808.10792*.

Joshi, A., Fidalgo, E., Alegre, E. & Fernández-Robles, L. (2023) ‘DeepSumm: Exploiting topic models and sequence-to-sequence networks for extractive text summarization’, *Expert Systems with Applications*, 211, p. 118442.

Solanki, S., Jain, S. & Bandhu, K.C. (2024) ‘Fusion of word embedding and encoder-decoder model for text summarization’, in *2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT)*. IEEE, pp. 1146–1150.

Kumar, S. & Solanki, A. (2023) ‘An abstractive text summarization technique using transformer model with self-attention mechanism’, *Neural Computing and Applications*, 35(25), pp. 18603–18622.

Sultana, M., Chakraborty, P. & Choudhury, T. (2022) ‘Bengali abstractive news summarization using Seq2Seq learning with attention’, in *Cyber Intelligence and Information Retrieval: Proceedings of CIIR 2021*. Springer Singapore, pp. 279–289.

Pan, H.X., Liu, H. & Tang, Y. (2019) ‘A sequence-to-sequence text summarization model with topic-based attention mechanism’, in *Web Information Systems and Applications: 16th International Conference, WISA 2019, Qingdao, China, September 20–22, 2019, Proceedings 16*. Springer International Publishing, pp. 285–297.