applied sciences

Review

Abstractive vs. Extractive Summarization: An Experimental Review

Nikolaos Giarelis [1,*], Charalampos Mastrokostas [2] and Nikos Karacapilidis [1]

[1] Industrial Management and Information Systems Lab, Department of Mechanical Engineering and Aeronautics, University of Patras, 26504 Rio Patras, Greece; karacap@upatras.gr
[2] Department of Electrical and Computer Engineering, University of Patras, 26504 Rio Patras, Greece; up1053708@upnet.gr
* Correspondence: giarelis@ceid.upatras.gr

Abstract: Text summarization is a subtask of natural language processing referring to the automatic creation of a concise and fluent summary that captures the main ideas and topics from one or multiple documents. Earlier literature surveys focus on extractive approaches, which rank the *top-n* most important sentences in the input document and then combine them to form a summary. As argued in the literature, the summaries of these approaches do not have the same lexical flow or coherence as summaries that are manually produced by humans. Newer surveys elaborate abstractive approaches, which generate a summary with potentially new phrases and sentences compared to the input document. Generally speaking, contrary to the extractive approaches, the abstractive ones create summaries that are more similar to those produced by humans. However, these approaches still lack the contextual representation needed to form fluent summaries. Recent advancements in deep learning and pretrained language models led to the improvement of many natural language processing tasks, including abstractive summarization. Overall, these surveys do not present a comprehensive evaluation framework that assesses the aforementioned approaches. Taking the above into account, the contribution of this survey is fourfold: (i) we provide a comprehensive survey of the state-of-the-art approaches in text summarization; (ii) we conduct a comparative evaluation of these approaches, using well-known datasets from the related literature, as well as popular evaluation scores such as *ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-LSUM, BLEU-1, BLEU-2* and *SACREBLEU*; (iii) we report on insights gained on various aspects of the text summarization process, including existing approaches, datasets and evaluation methods, and we outline a set of open issues and future research directions; (iv) we upload the datasets and the code used in our experiments in a public repository, aiming to increase the reproducibility of this work and facilitate future research in the field.

Keywords: text summarization; deep learning; language models; natural language processing; abstractive summarization; extractive summarization; empirical research; literature review

check for updates

Citation: Giarelis, N.; Mastrokostas, C.; Karacapilidis, N. Abstractive vs. Extractive Summarization: An Experimental Review. *Appl. Sci.* **2023**, *13*, 7620. https://doi.org/10.3390/app13137620

Academic Editor: Valentino Santucci

Received: 31 May 2023
Revised: 19 June 2023
Accepted: 26 June 2023
Published: 28 June 2023

1. Introduction

Due to the huge and continuously growing size of the textual corpora existing on the Internet, important information may go unnoticed or become lost. At the same time, the task of summarizing these resources by human experts is tedious and time consuming [1]. This necessitates the automation of the task. Natural language processing (NLP) is a multidisciplinary research field, merging aspects and approaches from computer science, artificial intelligence and linguistics; it deals with the development of processes that semantically and efficiently analyze vast amounts of textual data. Text summarization (TS) is a fundamental NLP subtask, which has been defined as the process of the automatic creation of a concise and fluent summary that captures the main ideas and topics of one or multiple documents [2]. When looking for related publications in the literature, we notice that the prevalent usage of the term TS refers to single document summarization, which is in the scope of this work; in any case, multi-document TS approaches also exist [3,4].

Diverse TS applications already exist, including: (i) literature summarization, aiming to handle a long document such as a book, a scientific article, or similar literature resources [5,6]; (ii) news outlet summarization, aiming to summarize information from one or multiple news outlet portals [7,8]; (iii) e-mail summarization [9,10]; (iv) legal document summarization, focusing on the extraction of important aspects from lengthy legal documents [11–13]; (v) social media summarization, where social media posts from multiple users are summarized in order to measure the social impact of a certain topic (this application is highly relevant to the field of opinion mining) [14,15]; (vi) argument summarization, as a means for meaningfully aggregating the public opinion in a digital democracy platform [16].

Recently published works outlined a series of insights about diverse TS characteristics. For instance, Ref. [17] assessed various techniques used by extractive summarization approaches, as well as the associated evaluation metrics; Ref. [18] presented an overview of TS datasets, approaches, and evaluation schemas; Ref. [4] elaborated on a comprehensive classification schema of TS approaches, based on their underlying techniques, and performed a comparative assessment of their performance through the use of various metrics; Ref. [19] focused on extractive TS approaches, evaluation metrics, and their limitations; Ref. [3] offered a comprehensive survey of TS and keyword extraction, the latter being a sibling task of TS. In addition, Refs. [2,20] are two comprehensive surveys of TS applications, approaches, datasets, and evaluation techniques, also reporting on associated limitations and challenges. Contrary to earlier works, Ref. [21] reported a comprehensive survey that focused only on abstractive TS approaches, taking into account recent deep learning approaches, while also presenting their comparative evaluation using various versions of the *ROUGE* metric [22].

The above works, however, have a series of limitations: (i) only a few of them [4,20,21] evaluated the approaches under consideration through a common evaluation framework (e.g., the *ROUGE* metric); (ii) only a few of them discussed deep learning approaches [2,20,21]; (iii) they did not use alternative evaluation metrics, which yield interesting results, e.g., *BLEU* [23], as discussed in [22–24]; (iv) they did not provide links to the code repositories of their experimental setups and datasets.

Several works reported the importance of developing a comprehensive evaluation framework [2,3,18,21]. Specifically, Ref. [2] stressed the need for the proposal of new solutions regarding the automatic evaluation of TS approaches, while [18] pointed out that the automatic evaluation of TS approaches remains a very promising research area with many open issues. Some of these issues include: (i) the lack of metrics that take into account the mismatch of synonymous terms between human assigned summaries and machine generated ones, (ii) the lack of datasets with quality summaries, and (iii) the lack of datasets for evaluation of multilingual approaches.

Taking into account the above remarks and overcoming the associated limitations, this paper aims to broaden and bring up to date our knowledge of the TS task. The overall contribution of this work is fourfold: (i) we thoroughly analyze state-of-the-art TS approaches, which build on recent advances in deep learning and language models and compare their basic techniques; (ii) we evaluate the performance of these approaches using popular evaluation scores, such as *ROUGE-1*, *ROUGE-2*, *ROUGE-L*, *ROUGE-LSUM*, *BLEU-1*, *BLEU-2*, and *SACREBLEU*, while comparing them to the best performing extractive approaches (as a baseline); (iii) we offer a set of insights on current TS approaches, and outline a set of open issues and future research directions; (iv) we make the code used for our experimental setup (https://github.com/cmastrokostas/Automatic_Text_Summarization (accessed on 19 June 2023)) and datasets (https://drive.google.com/drive/folders/1UJ_L5ZYYm52CQuURixc7DHs2h6Yz50F0?usp=sharing (accessed on 19 June 2023)) public, aiming to enhance the reproducibility of our work and facilitate future research efforts.

For the needs of this review, we used the search term “text summarization” along with the terms: “extractive”, “abstractive”, “method”, “approach”, “review”, “survey”, and “study”. Our search was conducted in Google Scholar, Elsevier Scopus, and the ACM

Digital Library. We narrowed down the findings of our search by only taking into account papers that have received more than 10 citations, or papers from a journal with an impact factor greater than 2.0. A detailed list of the sources (journals, conferences, workshops, repositories) of the 61 publications that satisfied these constraints and were analyzed in this work appears in Appendix A.

The remainder of this paper Is organized as follows. Background concepts and related work regarding extractive and abstractive TS approaches are thoroughly reviewed in Section 2. The evaluation of these approaches, as well as the datasets and metrics used, are presented in Section 3. Finally, concluding remarks, open issues, and future research directions are outlined in Section 4.

2. Related Work

Existing works have suggested various classification schemas of the TS approaches proposed so far [2–4,20]. The most prominent of them is based on the technique that creates the output summary [2]. According to it, the approaches can be divided into two major categories, namely extractive and abstractive ones. This section analyzes the selected TS approaches, which are divided into extractive (Section 2.1) and abstractive (Section 2.2). The evaluation datasets and metrics used in the TS literature are discussed in Sections 2.3 and 2.4, respectively.

2.1. Extractive Approaches

The goal of extractive approaches is to extract the most important sentences of the document under consideration. These are assembled into a concise summary that captures the most significant aspects of the original text. Various algorithms have been proposed for extractive summarization, each utilizing different techniques for the sentence ranking and extraction step, including: (i) statistical ones, which utilize statistical metrics such as word or sentence frequency; (ii) graph-based ones, which model the document into a graph of sentences, and then utilize graph theory concepts (e.g., centrality, community detection measures, etc.), and (iii) semantic-based ones, which model sentences and their terms into a co-occurrence matrix, which is then analyzed using distributional semantics [25]. In this context, this subsection discusses some of the most prominent approaches in extractive summarization, namely *Luhn*, *LSA*, *TextRank*, *LexRank, PositionRank,* and *TopicRank*.

Luhn [26] is one of the earliest approaches in extractive summarization. It utilizes statistical analysis to rank each sentence of a given text, based on the frequency of the most important words and their relative position in that sentence. The highest scoring sentences are extracted to form the final summary. However, this approach has a limitation, as it only focuses on individual words and does not consider the relationship between words or sentences.

Latent semantic analysis (*LSA*) was one of the earliest techniques used in an attempt to model the semantic relationships between words and capture key concepts in a document [27]. For the task of TS, the work of [28] proposed the *LSA* technique, which models a document as a term-sentence matrix that represents the frequency of each word in each sentence of the document. Then, it applies *singular value decomposition* (*SVD*) to extract the most important semantic features of the document in order to rank and extract the most important sentences. However, some drawbacks of this approach concern the dimensionality and the selection of sentences. To address them, Ref. [29] built a *semantic*-based approach using *LSA* that also used more advanced algorithms. Despite such improvements, summarization approaches built on *LSA* can be computationally expensive, especially for larger texts, due to the use of *SVD* [30].

The use of graph-based algorithms is another extractive summarization approach that addresses some limitations of earlier approaches since it performs fast and scalable summarizations. One of the earliest and most prominent graph-based ranking approaches is *TextRank* [31]. The first step of this approach is the representation of the document as a weighted graph of sentences. The sentences of the document are represented as nodes and the relationships between them as edges. A connection between two sentences indicates that there is similarity between them, measured as a function of their overlapping content. After the graph is created, the *PageRank* centrality algorithm [32] is applied to rank each sentence based on its connections to the other ones. Finally, the top-ranked sentences are selected to form a summary of the input document. The number of extracted sentences can be set as a user-defined parameter for the termination of the algorithm.

LexRank [33] is another graph-based algorithm that relies on *PageRank.* Its key difference is that each sentence is represented as a vector of the *TF-IDF* (*term frequency—inverse document frequency*) scores of the words it contains, while the relationship between these sentence vectors is measured using cosine similarity. A similarity matrix is created with each sentence represented as a row and column, and the elements of the matrix are computed as the cosine similarity score between the sentence vectors. Only similarities above a given threshold are included. To rank the sentences, *PageRank* is applied. The number of selected sentences can be set similarly to *TextRank.* Other graph-based approaches that build on *TextRank* are *TopicRank* [34] and *PositionRank* [35]. *TopicRank* uses a topic-modelling technique, which clusters sentences with similar topics and extracts the most important sentences of each cluster. *PositionRank* considers both the distribution of term positions in a text and the term frequencies in a biased *PageRank,* to rank sentences.

Many word embedding models have been developed since the introduction of the pioneering *Word2Vec* model [36]. Their goal is to capture semantic information for textual terms, thus increasing the accuracy of various NLP tasks. These embeddings are calculated for each term, and their mean vector representation is the document embedding. Recent advancements in deep learning allow the inference of sentence embeddings [37] from pretrained language models, while achieving better accuracy than earlier models. In this work, we utilize these sentence embeddings in an already proposed implementation of *LexRank* (https://www.sbert.net/examples/applications/text-summarization/README.html (accessed on 19 June 2023)). Our aim is to assess whether the introduction of sentence embeddings in the similarity step of the graph-based approach of *LexRank* improves the summarization accuracy of the base approach, when measured using the evaluation framework proposed in Section 3. The idea of incorporating word embeddings in extractive approaches is based on a number of works appearing in the literature [38–40].

2.2. Abstractive Approaches

The need for abstractive approaches resulted from a major drawback of extractive approaches, which is a lack of readability and coherence of the produced text, since extractive approaches utilize simple heuristics to extract and concatenate the most relevant sentences, without accounting for grammatical or syntactical rules [18]. To generate a fluent and coherent summary, more contextual information about the tokens of the input text is required, thus a family of models that generate new phrases in a similar manner to the paraphrasing process of a human reader is needed [2,4]. Many models for abstractive summarization have already been proposed in the literature. As seen in a recent survey [2], these include graph-based [41], rule-based [42], and semantic modelling [43] approaches. These earlier models, however, do not utilize recent advancements in deep learning, which improve many NLP tasks. Newer abstractive summarization approaches build on deep learning models, including: (i) the *convolutional neural networks* (*CNN*) and the *recurrent neural networks* (*RNNs*); (ii) *LSTM* and *GRU*, which improve the original *RNNs* and are discussed in [44]. Other neural architectures that are not based on *CNNs* and *RNNs* include *GAN* (generative adversarial networks). Certain works use these to build their abstractive approaches, as described in [45,46]. However, these yield lower evaluation scores (i.e., *ROUGE*) than recent deep learning models, which rely on the model explained in the next paragraph, as validated in [21].

Transformer [47] is a deep learning model that consists of a series of encoder and decoder layers, which utilize the attention mechanism to model the global dependencies of sequential data [48]. Specifically, the self-attention mechanism assigns different weights to different parts of the input, according to their contextual significance. These are encoded in hidden state layers when generating the output sequence. In addition, *Transformer* models use multi-head attention, which means that attention is applied in parallel to capture different patterns and relationships of the input data. *Transformer* uses the encoder-decoder model, which encodes information into hidden layers and then decodes it to generate output. These models are semisupervised, due to their unsupervised pretraining on large datasets, followed by supervised finetuning. Approaches built on this model achieve state-of-the-art performance on various text generation tasks, including abstractive summarization. Recent surveys [20,21] discussed and evaluated the differences between earlier abstractive approaches, including those that utilize deep learning models proposed before the introduction of the *Transformer* architecture. It is stressed here that the work presented in this article focuses on prominent pretrained language model approaches, which rely on the *Transformer* model and are discussed below.

T5 [49], which stands for *text-to-text transfer transformer*, is an approach that closely follows the *Transformer* architecture. It provides a general framework which converts multiple NLP tasks into sequential text-to-text ones. To address each task, it uses a task-specific prefix before the given sequence in the input. The pretraining process comprises both supervised and unsupervised training. The unsupervised objective of the approach includes masking random spans of tokens with unique sentinel tokens. The "corrupted" sentence is passed to the encoder, while the decoder learns to predict the dropped-out tokens on the output layer. A follow up approach, namely *mT5* [50], builds on *T5* to provide multilingual pretrained baseline models, which can be further finetuned to address diverse downstream tasks in multiple natural languages.

BART [51], which stands for *bidirectional auto-regressive transformers,* is a multitask deep learning approach, with abstractive summarization being included in them. *BART* utilizes a "denoising" autoencoder that learns the associations between a document and its "corrupted" form using various textual transformations. These include random token masking or deletion, text infilling, sentence permutation, and document rotation. This autoencoder is implemented as a sequence-to-sequence model with a bidirectional encoder and a left-to-right autoregressive decoder. For its pretraining, it optimizes a reconstruction loss (cross-entropy) function, where the decoder generates tokens found in the original document with higher probability.

PEGASUS [9], which stands for *pretraining with extracted gap-sentences for abstractive summarization*, is a deep learning approach pretrained solely for the downstream task of abstractive summarization. It introduces a novel pretraining objective for *Transformer*-based models, called *gap sentences generation* (*GSG*). This objective is specifically designed for the task of abstractive text summarization, as it involves the masking of whole sentences, rather than smaller text spans used in previous attempts. By doing so, it creates a "gap" in the input document, where the model is then trained to complete, by considering the rest of the sentences. Another key advantage of this approach is the selection of the masked sentences by utilizing a technique that ranks sentences based on their importance in the document rather than randomly, as suggested in earlier approaches.

Considering the rapidly increasing size and computational complexity of large pretrained models, as noted in [52], researchers were prompted to explore methods to compress them into smaller versions that maintain high accuracy and faster inference in terms of execution time. One such example is the work of [53] that proposes various comprehension techniques, including: (i) *direct knowledge distillation* (*KD*), which allows the knowledge transfer between a large model, referred to as the "teacher" model, into a smaller and "distilled" model, referred to as the "student" model; (ii) *pseudo-labels*, which replace the ground truth target documents of the student model with those of the teacher, and (iii) *shrink and finetune* (*SFT*), which shrinks the teacher model to student size by copying a subset of layers and then the finetuning student model again. They also provide various "distilled" pretrained model versions of large pretrained ones, produced by the *BART* and *PEGASUS* approaches.

2.3. Datasets

This subsection reports the selected evaluation datasets and their characteristics (an overview of them is given in Table 1). We opted for multiple datasets, aiming to test how well the approaches generalize over different data.

Table 1. Overview of the datasets, their statistics, and their source. The "#" symbol is used as an abbreviation for the term "number".

Dataset	Size	Mean #Words - Mean #Sentences (Text/Summary)	Link
CNN/Daily Mail	312 k News Articles	766/53 words—29.74/3.72 sentences	https://github.com/abisee/cnn-dailymail (accessed on 19 June 2023) https://www.kaggle.com/datasets/gowrishankarp/newspaper-text-summarization-cnn-dailymail (accessed on 19 June 2023)
XSum	227 k News Articles	431.07/23.26 words—19.77/1.00 sentence	https://github.com/EdinburghNLP/XSum (accessed on 19 June 2023)
XLSum (*English*)	330 k News Articles	460.42/22.07 words—23.54/1.11 sentences	https://github.com/csebuetnlp/xl-sum/tree/master/seq2seq (accessed on 19 June 2023)
SAMSum	16 k Chat Dialogues	93.77/20.3 words—19.26/3.07 sentences	https://arxiv.org/src/1911.12237v2/anc/corpus.7z (accessed on 19 June 2023)
ddit TIFU (*TL;DR*)	123 k Reddit Posts (42 K Posts)	444/23 words—22/1.4 sentences-	https://github.com/ctr4si/MMN (accessed on 19 June 2023)
BillSum (*US*)	22 k Legislation Bills	1686/243 words—42/7.1 sentences	https://github.com/FiscalNote/BillSum (accessed on 19 June 2023)

CNN/Daily Mail [54] is a dataset containing over 300,000 news articles from *CNN* and the *Daily Mail* newspaper, written between 2007 and 2015. This dataset is distributed in three major versions. The first one was made for the NLP task of question answering and contains 313 k unique news articles and close to 1 M questions. The second version was restructured for the task of TS; the data in this version are anonymized. The third version provides a nonanonymized version of the data, where individuals' names can be found in the dataset. Each article is accompanied by a list of bullet point summaries, which abstractively summarize a key aspect of the article. The *CNN/Daily Mail* dataset has 3 splits: training (92%, 287,113 articles), validation (4.3% 13,368 articles), and test (3.7%, 11,490 articles).

XSum (standing for *e**X**treme **Sum**marization*) is a dataset that provides over 220,000 *BBC* news articles covering various topics [8]. Each article is accompanied by a one-sentence summary written by a human expert, who for the most part was the original author of the article. *XSum* has 3 splits: training (90%, 204,045 articles), validation (5%, 11,332 articles), and test (5%, 11,334 articles).

SAMSum [55] is a dataset that contains more than 16,000 online chat conversations written by linguists. These conversations cover diverse topics and formality styles including emoticons, slang words, and even typographical errors. They are also annotated with short third-person summaries, explaining the dialogue between different people.

Reddit TIFU [56] is a dataset consisting of 123,000 Reddit posts from the */r/tifu* online discussion forum. These posts are informal stories that include a short summary, which is the title of the post, and a longer one, known as the "*TL;DR*" (*too long; didn't read*) summary.

BillSum [57] is a dataset that deals with US Congressional (USC) and California (CA) state bill summarization. This corpus contains legislation documents from five to twenty thousand characters. In total, it contains 22,200 USC (18,949 train documents and 3269 test documents) and 1200 CA state bills (1237 test documents), accompanied by summaries written by human experts. The data are collected from the US Publishing Office *Govinfo* and the CA legislature's website.

XLSum [58] is a multilingual dataset that covers 44 different languages and offers more than 1 million pairs of BBC news articles and their summaries. Since our review focuses on summarization approaches that were trained and finetuned in the English language, we utilize the English part of the dataset which has 3 splits: training (93%, 306,522 articles), validation (3.5%, 11,535 articles), and test (3.5%, 11,535 articles). The authors of *XLSum* argue that they provide a more concise dataset, with less irrelevant summaries compared to *CNN/Daily Mail* and *XSum* datasets.

Regarding the statistics found in Table 1, we collected them from either the original publications mentioned above or the summarization survey appearing in [59] for the case of Reddit *TIFU* and *BillSum*. For the cases of *SAMSum* and *XLSum*, we run our own analysis since no statistics for these datasets were available in the literature.

2.4. Evaluation Metrics

For evaluation purposes, we employ two well-known families of metrics, namely *BLEU* and *ROUGE*. In general, both assess the number of matching *n-grams* (sequence of terms) between the machine generated summary and the human assigned one. *BLEU* [23], standing for *bilingual evaluation understudy*, is an automatic evaluation method that was originally created for the task of machine translation, but can also be applied for the automatic text summarization task, as suggested in [22–24]. It is based on the precision metric, which measures the number of words from the machine-generated (candidate) sentence that match the words in the human-written (reference) sentence, divided by the total number of words in the candidate sentence. Specifically, it utilizes a brevity penalty for short sentences and a modified precision. This precision calculates the geometric average of *n-gram* precisions, while penalizing word repetition. Two variations of BLEU, namely *BLEU-1* and BLEU-2, use unigram precision and both unigram and bigram precisions, respectively. In addition, we included another variation of the method, called *SACREBLEU* [60]. This metric tackles the problem of nonreproducibility in academic papers utilizing the original *BLEU* method. This occurs due to differences in parametric setups for *BLEU*, which results in different scores across publications.

ROUGE [22], which stands for *recall-oriented understudy for gisting evaluation*, was inspired by the success of the *n-gram* overlap measure utilized by *BLEU*. In contrast to *BLEU*, *ROUGE* was introduced as a recall-oriented metric. Specifically, the most common setups are: (i) *ROUGE-1* and *ROUGE-2* for unigrams and bigrams, respectively; (ii) *ROUGE-L*, which utilizes the *longest common subsequence* (*LCS*) between the reference and the machine-generated summary.It is noted that a variation of *ROUGE-L*, called *ROUGE-LSUM*, is computed on the summary-level (contrary to *ROUGE-L* that is computed on the sentence-level).

3. Evaluation

This section reports the experimentations carried out to evaluate the approaches presented in Section 2. Specifically, Section 3.1 provides a comprehensive overview of the hardware and software used in this research. Section 3.2 outlines the experimental setup. It is noted that we evaluated the selected automatic summarization methods using the Python programming language. Our experiments incorporate a wide variety of extractive and abstractive models, compared through the same evaluation framework across heterogenous datasets.

3.1. Hardware and Software Specifications

We ran the experiments on a computer using one Nvidia RTX 3060 12 GB VRAM GPU, a Ryzen 5 6-core CPU, and 16 GB of RAM. We used the GPU to speed up the abstractive summarization process, since popular machine learning (ML) libraries (e.g., *PyTorch*) offer a significant speedup in terms of execution time, when using one or multiple GPU devices for parallelism. With respect to software specifications, we utilized extractive approaches from the libraries of *sumy* (https://miso-belica.github.io/sumy/ (accessed on 19 June 2023)) and *pyTextRank* (https://derwen.ai/docs/ptr/ (accessed on 19 June 2023)) [61]; for abstractive approaches, we used various *Transformer*-based models [62] from *Huggingface* (https://huggingface.co/ (accessed on 19 June 2023)). All these approaches and their implementations are listed in Table 2.

Table 2. Extractive and Abstractive Summarization Approaches.

Type	Approach (Models)	Link
Extractive Approaches	*PositionRank* (*PositionRank*)	https://github.com/DerwenAI/pytextrank/blob/main/pytextrank/positionrank.py (accessed on 19 June 2023)
	TextRank (*pyTextRank, sumy*)	https://github.com/DerwenAI/pytextrank/blob/main/pytextrank/base.py (accessed on 19 June 2023) https://github.com/miso-belica/sumy/blob/main/sumy/summarizers/text_rank.py (accessed on 19 June 2023)
	TopicRank (*TopicRank*)	https://github.com/DerwenAI/pytextrank/blob/main/pytextrank/topicrank.py (accessed on 19 June 2023)
	LexRank (*LexRank*)	https://github.com/miso-belica/sumy/blob/main/sumy/summarizers/lex_rank.py (accessed on 19 June 2023)
	LSA (*LSA*)	https://github.com/miso-belica/sumy/blob/main/sumy/summarizers/lsa.py (accessed on 19 June 2023)
	Luhn (*Luhn*)	https://github.com/miso-belica/sumy/blob/main/sumy/summarizers/luhn.py (accessed on 19 June 2023)
Abstractive Approaches	*BART* (*BART-large-XSum, BART-large-CNN, BART-large-CNN-SAMSum*)	https://huggingface.co/facebook/bart-large-xsum (accessed on 19 June 2023) https://huggingface.co/facebook/bart-large-cnn (accessed on 19 June 2023) https://huggingface.co/philschmid/bart-large-cnn-samsum (accessed on 19 June 2023)
	mT5 (*mT5-multilingual-XLSum*)	https://huggingface.co/csebuetnlp/mT5_multilingual_XLSum (accessed on 19 June 2023)
	PEGASUS (*PEGASUS-large, PEGASUS-multi_news, PEGASUS-Xsum, PEGASUS-CNN_dailymail*)	https://huggingface.co/google/pegasus-large (accessed on 19 June 2023) https://huggingface.co/google/pegasus-multi_news (accessed on 19 June 2023) https://huggingface.co/google/pegasus-xsum (accessed on 19 June 2023) https://huggingface.co/google/pegasus-cnn_dailymail (accessed on 19 June 2023)
	DistilBART (*distilBART-CNN-12-6, distilBART-Xsum-12-6*)	https://huggingface.co/sshleifer/distilbart-cnn-12-6 (accessed on 19 June 2023) https://huggingface.co/sshleifer/distilbart-xsum-12-6 (accessed on 19 June 2023)

3.2. Experimental Setup and Evaluation Results

This subsection describes the models, datasets, and metrics used in our experiments. Regarding the models, we utilized pretrained models obtained from *Huggingface*. Our experiments mainly focused on monolingual, single-document summarization datasets and methods. We selected publicly available evaluation datasets covering different text summarization topics to assess how well the selected models generalize on diverse datasets in terms of size, domain, and topics. In Section 3.2.1, we describe the various deep learning setups used by the abstractive models; for the extractive models, we used the respective implementations from the *sumy* and *pyTextRank* libraries [61], with their default hyper-parameters. These predefined hyperparameters were set according to the state-of-the-art literature. The objective of our evaluation framework was to check which model performs better, whether extractive or abstractive; particularly for abstractive models, we wanted to check their ability to generalize even in the case of missing training or finetuning data.

For the needs of our evaluation, we restructured the chosen datasets to create a uniform directory structure containing simple *txt* files. For the *CNN/Daily Mail* dataset, we used its nonanonymized version, and specifically the test split mentioned in Table 1; this version structures the original test data into a *csv* format (the original data can be found in this link (https://github.com/abisee/cnn-dailymail (accessed on 19 June 2023)). We also utilized the original tests splits of *XSum* and *SAMSum*, the original English test split of *XLSum*, and the original US test data from *BillSum*. To the best of our knowledge, there are no official data splits for *Reddit TIFU*. Therefore, for our evaluation purposes we choose 5% of the document from the long summary version of this dataset. Also, before converting and splitting the data to the desired format for this dataset, we preprocessed them to remove posts with missing summaries, since the inclusion of the longer summary is optional by the users.

Regarding the evaluation metrics, we used the following implementations: (i) for the *ROUGE* metric, we used the implementation described in this link (https://huggingface.co/spaces/evaluate-metric/rouge (accessed on 19 June 2023)); (ii) for the *BLUE* metric, we used the implementation described in this link (https://huggingface.co/spaces/evaluate-metric/bleu (accessed on 19 June 2023)); (iii) for the *SACREBLUE* metric, we used the implementation described in this link (https://huggingface.co/spaces/evaluate-metric/sacrebleu (accessed on 19 June 2023)).

Considering the length of the generated summaries, for the extractive approaches it is required to set the desired number of output sentences. For the abstractive models, the output length can be configured by tuning the sequence generation parameters (*min_length*, *max_length*) of each model. In the abstractive models, the number of extracted sentences cannot be set, as opposed to the number of generated tokens. The disadvantage of manually adjusting this setting is that the model may abruptly end a sentence before the period character. For the abstractive approaches, we considered the default hyperparameters set by the original authors.

3.2.1. Abstractive Models

In this section, we describe the *Transformer*-based models and setups that appear in Table 2 and are based on the abstractive approaches presented in Section 2. All models set the dimension for the input layer to 1024, the only exception being *mT5-multilingual-XLSum* for which it is set to 768. Specifically:

- *BART-large-CNN* is a finetuned version of the large sized model of *BART* (*BART-large*) on *CNN/Daily Mail*. The *BART-large* model uses the same number (i.e., 12) for each of the encoder, decoder, and hidden layers. The maximum sequence length (*max_position_embeddings*) of the model is 1024 tokens, while the output lengths (*min_length*, *max_length*) are set to (56, 142) tokens.
- *BART-large-XSum* is a finetuned version of the large sized model of *BART* on *XSum*. It uses the same number of layers as the previous model. The max sequence length (*max_position_embeddings*) of the model is 1024 tokens, while the output lengths (*min_length*, *max_length*) are set to (11, 62) tokens.
- *BART-large-CNN-SAMSum* uses the *BART-large-CNN* model; it is further finetuned on examples from the *SAMSum* dataset. It has the same number of layers, maximum sequence, and output lengths as the *BART-large-CNN* model.
- *PEGASUS-large* is actually the "*PEGASUS*$_{LARGE}$ (*mixed, stochastic*)" model reported in [9]. The key difference of this model is that it was pretrained on two large text corpora, namely *C4* and *HugeNews*, with a different training configuration compared to the other PEGASUS models reported in [9]. This model features an updated tokenizer and an enhanced training process. The model has an encoder, a decoder, and a hidden state, each consisting of 16 layers. The max sequence length (*max_position_embeddings*) of the model is 1024 tokens, while the maximum output length (*max_length*) is set to 256 tokens.
- *PEGASUS-multi_news* uses the *PEGASUS-large* model finetuned on the *Multi-News* [63] dataset, which was originally made for the task of multi-document summarization. It has the same number of maximum sequence and output lengths.
- *PEGASUS-XSum* uses the *PEGASUS-large* model finetuned on *XSum*. It has a maximum sequence length (*max_position_embeddings*) of 512 tokens and an output length limit (*max_length*) of 64 tokens.

- *PEGASUS-CNN_dailymail* uses the *PEGASUS-large* model finetuned on *CNN/Daily Mail*. It has a maximum sequence length (*max_position_embeddings*) of 1024 tokens and an output length limit (*max_length*) of 128 tokens.
- *DistilBART-CNN-12-6* is a student model of *BART* that is trained for *CNN/Daily Mail* using the distillation technique of KD. It features an encoder and a hidden state, each consisting of 12 layers, and a decoder consisting of 6 (half of the teacher model's decoder layers). The student model has a maximum sequence length (*max_position_embeddings*) of 1024 tokens, while output length limits (*min_length*, *max_length*) are set to (56, 142) tokens.
- *DistilBART-XSum-12-6* is a student model of *BART* that is trained for the *XSum* dataset using the same distillation technique as the previous model. It features an encoder and a hidden state with 12 layers, and a decoder with 6 of the teacher model's layers. The student model has a maximum sequence length (*max_position_embeddings*) of 1024 tokens, while output length limits (*min_length*, *max_length*) are set to (11, 62) tokens.
- *mT5-multilingual-XLSum* is a model that uses Google's *mT5* (multilingual-T5) base model finetuned on the *XLSum* dataset for the task of multilingual summarization [55]. The model has an encoder, a decoder, and a hidden state, each consisting of 12 layers. The original authors truncated the input sequence to 512 tokens, while the maximum output length (*max_length*) is set to 84 tokens.

3.2.2. Evaluation Results

Tables 3–8 summarize the evaluation results of our experiments. For this process, we used the test splits of the datasets mentioned in Section 3.2.1. To evaluate the performance of the selected approaches, we compared their machine-generated summary with the corresponding reference one. This was executed by utilizing the metrics presented in Section 2.4. For each metric, we calculated the macro (mean) score of the scores computed on each dataset. For the pretrained models, we assess their performance without further finetuning on the evaluation datasets. Our aim is to measure the impact of different learning parameters and datasets of distinct pretrained models belonging to the same approach.

Table 3. Experimental Evaluation Results on *CNN/Daily Mail*.

	Model	R1	R2	RL	RLSum	B1	B2	SB
Extractive	*PositionRank*	**0.3341**	**0.1277**	**0.2130**	**0.2819**	**0.2507**	**0.1432**	**0.0805**
	LexRank	0.3245	0.1146	0.2013	0.2665	0.2334	0.1252	0.0693
	e-LexRank	0.3211	0.1171	0.1983	0.2679	0.2323	0.1276	0.0715
	TextRank (*pyTextRank*)	0.3126	0.1118	0.1940	0.2615	0.2310	0.1255	0.0702
	TextRank (*sumy*)	0.2827	0.0956	0.1782	0.2339	0.1895	0.0994	0.0544
	Luhn	0.3091	0.1160	0.1981	0.2584	0.2149	0.1209	0.0684
	TopicRank	0.2987	0.1170	0.2022	0.2712	0.2383	0.1311	0.0728
	LSA	0.2933	0.0909	0.1816	0.2399	0.2161	0.1046	0.0593
Abstractive	*distilBART-CNN-12-6*	**0.4292**	**0.2077**	0.2971	0.3646	**0.3625**	**0.2384**	0.1420
	BART-large-CNN	0.4270	0.2058	**0.2977**	**0.3647**	0.3156	0.2052	**0.1433**
	PEGASUS-CNN_dailymail	0.4186	0.2025	0.2983	0.3607	0.3259	0.2124	0.1312
	BART-large-CNN-SAMSum	0.4203	0.1945	0.2904	0.3560	0.3081	0.1963	0.1303
	PEGASUS-large	0.3300	0.1330	0.2294	0.2803	0.2341	0.1367	0.0807
	PEGASUS-multi_news	0.2794	0.1042	0.1714	0.2233	0.1624	0.0890	0.0449
	BART-large-XSum	0.2413	0.0737	0.1635	0.2099	0.0834	0.0385	0.0242
	distilBART-XSum-12-6	0.2197	0.0624	0.1518	0.1919	0.0630	0.0273	0.0177
	mT5-multilingual-XLSum	0.2017	0.0545	0.1444	0.1789	0.0521	0.0218	0.1499
	PEGASUS-XSum	0.1998	0.0672	0.1398	0.1749	0.0749	0.0381	0.0242

Table 4. Experimental Evaluation Results on *XSum*.

	Model	R1	R2	RL	RLSum	B1	B2	SB
Extractive	*TopicRank*	**0.1828**	**0.0258**	**0.1313**	**0.1313**	**0.1246**	0.0266	**0.0266**
	TextRank (*sumy*)	0.1738	0.0248	0.1266	0.1266	0.1196	**0.0271**	0.0227
	TextRank (*pyTextRank*)	0.1646	0.0219	0.1197	0.1197	0.1125	0.0227	0.0248
	Luhn	0.1734	0.0246	0.1254	0.1254	0.1193	0.0265	0.0244
	e-LexRank	0.1725	0.0243	0.1256	0.1256	0.1147	0.0239	0.0259
	LexRank	0.1696	0.0226	0.1237	0.1237	**0.1149**	0.0238	0.0257
	PositionRank	0.1618	0.0184	0.1188	0.1188	0.1121	0.0200	0.0247
	LSA	0.1518	0.0165	0.1101	0.1101	0.1047	0.0165	0.0250
Abstractive	*PEGASUS-XSum*	**0.4573**	**0.2405**	**0.3840**	**0.3840**	**0.3484**	**0.2374**	**0.1602**
	BART-large-XSum	0.4417	0.2194	0.3649	0.3649	0.3445	0.2250	0.1470
	distilBART-XSum-12-6	0.4397	0.2197	0.3665	0.3665	0.3350	0.2204	0.1449
	mT5-multilingual-XLSum	0.3520	0.1386	0.2845	0.2845	0.2536	0.1394	0.0889
	BART-large-CNN-SAMSum	0.2084	0.0435	0.1399	0.1399	0.1251	0.0429	0.0231
	distilBART-CNN-12-6	0.1985	0.0357	0.1307	0.1307	0.1122	0.0348	0.0199
	PEGASUS-CNN_dailymail	0.1979	0.0356	0.1339	0.1339	0.1247	0.0368	0.0209
	BART-large-CNN	0.1972	0.0339	0.1303	0.1303	0.1167	0.0340	0.0197
	PEGASUS-large	0.1654	0.0266	0.1146	0.1146	0.0988	0.0252	0.0190
	PEGASUS-multi_news	0.1578	0.0491	0.1114	0.1114	0.0812	0.0387	0.0173

Table 5. Experimental Evaluation Results on *XLSum* (English).

	Model	R1	R2	RL	RLSum	B1	B2	SB
Extractive	*TopicRank*	**0.1900**	**0.0279**	**0.1354**	**0.1354**	**0.1291**	**0.0284**	**0.0271**
	e-LexRank	0.1841	0.0277	0.1325	0.1325	0.1224	0.0269	0.0267
	Luhn	0.1812	0.0259	0.1293	0.1293	0.1225	0.0275	0.0240
	TextRank (*sumy*)	0.1802	0.0260	0.1304	0.1304	0.1222	0.0278	0.0232
	TextRank (*pyTextRank*)	0.1690	0.0230	0.1224	0.1224	0.1152	0.0229	0.0249
	LexRank	0.1778	0.0250	0.1290	0.1290	0.1207	0.0258	0.0265
	PositionRank	0.1627	0.0194	0.1188	0.1188	0.1118	0.0204	0.0246
	LSA	0.1526	0.0169	0.1101	0.1101	0.1057	0.0165	0.0249
Abstractive	*PEGASUS-XSum*	**0.4343**	**0.2189**	**0.3617**	**0.3617**	**0.3269**	**0.2148**	**0.1452**
	distilBART-XSum-12-6	0.4237	0.2048	0.3500	0.3500	0.3208	0.2041	0.1362
	BART-large-XSum	0.4223	0.2009	0.3450	0.3450	0.3260	0.2055	0.1347
	mT5-multilingual-XLSum	0.3623	0.1491	0.2927	0.2927	0.2586	0.1462	0.0946
	BART-large-CNN-SAMSum	0.2097	0.0441	0.1406	0.1406	0.1239	0.0417	0.0227
	BART-large-CNN	0.2011	0.0348	0.1322	0.1322	0.1170	0.0333	0.0199
	distilBART-CNN-12-6	0.1999	0.0355	0.1311	0.1311	0.1112	0.0335	0.0196
	PEGASUS-CNN_dailymail	0.1966	0.0339	0.1318	0.1318	0.1215	0.0342	0.0197
	PEGASUS-large	0.1694	0.0288	0.1158	0.1158	0.0970	0.0325	0.0177
	PEGASUS-multi_news	0.1443	0.0422	0.1014	0.1014	0.0717	0.0261	0.0144

Table 6. Experimental Evaluation Results on *Reddit TIFU*.

	Model	**R1**	**R2**	**RL**	**RLSum**	**B1**	**B2**	**SB**
Extractive	*TopicRank*	**0.1770**	**0.0284**	**0.1290**	**0.1290**	**0.1195**	0.0295	**0.0270**
	Luhn	0.1724	0.0284	0.1233	0.1233	0.1190	**0.0311**	0.0255
	e-LexRank	0.1703	0.0265	0.1264	0.1264	0.1111	0.0260	0.0268
	TextRank (*sumy*)	0.1689	0.0263	0.1215	0.1215	0.1164	0.0290	0.0239
	TextRank (*pyTextRank*)	0.1534	0.0228	0.1145	0.1145	0.1036	0.0237	0.0246
	LexRank	0.1673	0.0250	0.1223	0.1223	0.1131	0.0261	0.0262
	LSA	0.1474	0.0179	0.1095	0.1095	0.0994	0.0172	0.0248
	PositionRank	0.1304	0.0174	0.0990	0.0990	0.0864	0.0169	0.0229
Abstractive	*BART-large-CNN-SAMSum*	**0.1834**	**0.0421**	**0.1305**	**0.1305**	0.1065	**0.0386**	0.0228
	BART-large-XSum	0.1676	0.0329	0.1274	0.1274	**0.1088**	0.0300	**0.0315**
	distilBART-XSum-12-6	0.1697	0.0314	0.1300	0.1300	0.1059	0.0283	0.0313
	distilBART-CNN-12-6	0.1657	0.0340	0.1143	0.1143	0.0924	0.0315	0.0184
	PEGASUS-large	0.1617	0.0295	0.1133	0.1133	0.0993	0.0301	0.0203
	PEGASUS-CNN_dailymail	0.1596	0.0326	0.1118	0.1118	0.0956	0.0316	0.0169
	BART-large-CNN	0.1570	0.0328	0.1088	0.1088	0.0883	0.0304	0.0167
	PEGASUS-XSum	0.1428	0.0228	0.1135	0.1135	0.0779	0.0167	0.0260
	PEGASUS-multi_news	0.1063	0.0243	0.0743	0.0743	0.0524	0.0197	0.0095

Table 7. Experimental Evaluation Results on *SAMSum*.

	Model	**R1**	**R2**	**RL**	**RLSum**	**B1**	**B2**	**SB**
Extractive	*e-LexRank*	**0.2762**	**0.0769**	**0.2059**	**0.2059**	**0.1054**	**0.0525**	**0.0393**
	LexRank	0.1654	0.0310	0.1136	0.1136	0.0979	0.0309	0.0183
	TopicRank	0.1613	0.0313	0.1112	0.1112	0.0958	0.0316	0.0180
	LSA	0.1622	0.0253	0.1101	0.1101	0.0987	0.0262	0.0173
	TextRank (*pyTextRank*)	0.1570	0.0287	0.1063	0.1063	0.0928	0.0282	0.0172
	TextRank (*sumy*)	0.1471	0.0296	0.1027	0.1027	0.0823	0.0282	0.0154
	PositionRank	0.1557	0.0272	0.1073	0.1073	0.0949	0.0270	0.0178
	Luhn	0.1508	0.0306	0.1049	0.1049	0.0850	0.0292	0.0161
Abstractive	*BART-large-CNN-SAMSum*	**0.4004**	**0.2007**	**0.3112**	**0.3112**	**0.2603**	**0.1742**	**0.1165**
	BART-large-CNN	0.3041	0.1009	0.2272	0.2272	0.1487	0.0793	0.0508
	PEGASUS-CNN_dailymail	0.2866	0.0833	0.2220	0.2220	0.1433	0.0697	0.0447
	distilBART-CNN-12-6	0.2884	0.0937	0.2149	0.2149	0.1350	0.0697	0.0466
	BART-large-XSum	0.2572	0.0514	0.1878	0.1878	0.1542	0.0460	0.0431
	PEGASUS-large	0.2589	0.0628	0.2021	0.2021	0.1008	0.0468	0.0429
	distilBART-XSum-12-6	0.2121	0.0328	0.1561	0.1561	0.1276	0.0275	0.0361
	mT5-multilingual-XLSum	0.1787	0.0235	0.1359	0.1359	0.1082	0.0190	0.0313
	PEGASUS-XSum	0.1428	0.0228	0.1135	0.1135	0.0779	0.0167	0.0260
	PEGASUS-multi_news	0.1127	0.0200	0.0804	0.0804	0.0543	0.0149	0.0103

Table 8. Experimental Evaluation Results on *BillSum*.

	Model	R1	R2	RL	RLSum	B1	B2	SB
Extractive	*LexRank*	**0.3845**	**0.1888**	**0.2460**	**0.2460**	**0.2663**	**0.1815**	**0.1140**
	TextRank (*pyTextRank*)	0.3638	0.1735	0.2168	0.2168	0.2477	0.1651	0.1029
	TextRank (*sumy*)	0.3503	0.1799	0.2320	0.2320	0.2437	0.1629	0.1033
	PositionRank	0.3601	0.1686	0.2153	0.2153	0.2436	0.1606	0.0996
	e-LexRank	0.3595	0.1750	0.2205	0.2205	0.2397	0.1620	0.1032
	TopicRank	0.3568	0.1763	0.2174	0.2174	0.2334	0.1662	0.1036
	Luhn	0.3521	0.1812	0.2342	0.2342	0.2349	0.1640	0.1041
	LSA	0.3480	0.1406	0.2115	0.2115	0.2288	0.1380	0.0846
Abstractive	*PEGASUS-large*	**0.3568**	0.1480	0.2268	0.2268	**0.2312**	**0.1400**	**0.0901**
	BART-large-CNN-SAMSum	0.3186	**0.1571**	**0.2301**	**0.2301**	0.1333	0.0881	0.0562
	distilBART-CNN-12-6	0.3025	0.1465	0.2161	0.2161	0.1303	0.0836	0.0528
	PEGASUS-CNN_dailymail	0.2962	0.1405	0.2116	0.2116	0.1311	0.0832	0.0543
	BART-large-CNN	0.2954	0.1433	0.2146	0.2146	0.1168	0.0746	0.0495
	PEGASUS-multi_news	0.2801	0.0721	0.1661	0.1661	0.1909	0.0864	0.0406
	mT5-multilingual-XLSum	0.1486	0.0584	0.1155	0.1155	0.0232	0.0132	0.0081
	PEGASUS-XSum	0.1440	0.0742	0.1171	0.1171	0.0264	0.0152	0.0129
	BART-large-XSum	0.1327	0.0641	0.0988	0.0988	0.0159	0.0093	0.0070
	distilBART-XSum-12-6	0.1237	0.0540	0.0960	0.0960	0.0157	0.0091	0.0065

As mentioned earlier, the extractive approaches require a specification of the desired number of extracted sentences. To facilitate the fairness of the evaluation, we set this number across all extractive approaches, taking into account the sentence length of the summaries of each specific dataset. For *CNN/Daily Mail*, we set that number to three sentences, due to the average summary length of the dataset. For *XSum*, we set this number to one, since its reference summaries comprise just a single sentence. XLSum has similar summaries, therefore, we used the same setting as *XSum.* For *Reddit TIFU*, we set the number of sentences for the extractive approaches to one, due to the results of our analysis. In the case of *BillSum,* we kept the default output length setting for the extractive approaches, which is four sentences. To evaluate the extractive approaches on *SAMSum*, we set the number of extracted sentences to three, due to the results of our analysis.

In the following tables, **R1**, **R2**, **RL**, and **RLSUM** stand for the macro scores of **ROUGE-1**, **ROUGE-2**, **ROUGE-L**, and **ROUGE-LSum**, respectively. Similarly, **B1**, **B2**, and **SB** stand for the macro scores of **BLEU-1**, **BLEU-2**, and **SACREBLEU**, respectively. We bolded the best scoring approaches among both the extractive and the abstractive approaches. We also underlined the second and third best scoring abstractive approaches.

As shown in Table 3, the top abstractive models scored higher than the extractive ones on *CNN/Daily Mail*. Specifically, *distilBART-CNN-12-6* had the best **R1**, **R2**, **B1**, and **B2** scores, while *BART-large-CNN* had the best **RL**, **RLSum**, and **SB** scores. *PEGASUS-CNN_dailymail* and *BART-large-CNN-SAMSum,* which were also finetuned on this dataset, had slightly lower scores. Most abstractive models, which were not finetuned on *CNN/Daily Mail*, demonstrated lower scores compared to the extractive ones. Among the extractive approaches, *PositionRank* scored the best across all evaluation metrics. Among the two *TextRank* variations, the implementation from *pyTextRank* performed better on this dataset. *e-LexRank* is the variation of *LexRank* that uses sentence transformers embeddings, which did not perform better than *LexRank*.

As shown in Tables 4 and 5, the top abstractive models scored much higher than the extractive ones on *XSum*. Specifically, *PEGASUS-XSum* scored the best across all metrics, followed by *BART-large-XSum* and *distilBART-x*sum-12-6. The *mT5-multilingual-XLSum* model has been finetuned on the multilingual *XLSum* dataset, which contains similar data in to *XSum* in its English part [58]. Most abstractive models, which are not finetuned on *XSum*, had higher results than the extractive ones, apart from *PEGASUS-large* and *PEGASUS-multi_news,* on most metrics. Among the extractive approaches, *TopicRank* scored the best on every metric except for **B2,** where *TextRank* (*sumy*) received the best score. Among the two *TextRank* variations, the implementation from *sumy* performed better on this dataset. *e-LexRank* performed slightly better than *LexRank* in both Tables 4 and 5; however, it was not the top performing extractive approach.

As specified in Section 3.2.1, we are conducting the experimental evaluation on the English part of *XLSum*, which is highly similar to *XSum*. As shown in Tables 4 and 5, the abstractive models that are finetuned on *XSum* have similar or even better performance than the *mT5-multilingual-XLSum* model. We believe that this better performance of the *XSum* models is achieved due to the fact that they were solely trained on English corpora, unlike *mT5-multilingual-XLSum*, which was trained on a multilingual corpus. Among the extractive approaches, *TopicRank* scored the best across all the evaluation metrics. Among the two *TextRank* variations, the implementation from *sumy* performed better across all metrics.

As shown in Table 6, the abstractive models had comparable results with the extractive ones on *Reddit TIFU*, with *BART-large-CNN-SAMSum* having the best score across all evaluation metrics except **B1** and **SB** where *TopicRank* and *BART-large-XSum* performed better, respectively. Among the two *TextRank* variations, the implementation from *sumy* performed better on this dataset. *e-LexRank* performed better than *LexRank*, however it was not the best performing extractive approach.

As shown in Table 7, most abstractive models outperformed the extractive ones in the case of *SAMSum*. The *BART-large-CNN-SAMSum* performed significantly better across all evaluation metrics, due to its additional finetuning on this specific dataset. The rest of the abstractive models had higher but comparable results to the extractive ones, apart from *PEGASUS-multi_news*. All extractive approaches had comparable results, with e-*LexRank* achieving the best performance across most metrics. Among the two *TextRank* variations, the implementation from *pyTextRank* performed better across all metrics, with the exception of **R2**.

As shown in Table 8, all abstractive models exhibited lower results than the extractive ones in most metrics in the case of *BillSum*. *LexRank* scored better across all metrics. Among the abstractive approaches, *PEGASUS-large* had the best performance in **R1**, **B1**, **B2**, and **SB**, while *BART-large-CNN-SAMSum* performed better in **R2** and **RL**. The extractive approaches had comparable results, with *LexRank* being the best model for this dataset. Among the two *TextRank* variations, the implementation from *pyTextRank* performed better on **R1**, **B1**, and **B2,** while the *sumy* implementation performed better on **R2**, **RL**, and **SB**. *e-LexRank* performed worse than *LexRank* in Table 8.

4. Conclusions, Open Issues, and Future Work

This paper has thoroughly evaluated a set of prominent extractive and abstractive summarization approaches on heterogeneous datasets, aiming to comparatively access them and point out their strengths and weaknesses. Our experimentations reveal the following set of conclusions:

- Abstractive models that are finetuned on data similar to the test data produce significantly better results. The rest of the abstractive models, which are finetuned on data different than the test data, in most cases perform similarly or even better than the extractive ones. There is no clear winner among the *BART*, *PEGASUS,* and *DistilBART* approaches since none of them outperformed the others across diverse datasets. However, it is worth noting that in some cases the *DistilBART* student outperformed the *BART* teacher model.
- Regarding the extractive approaches, their evaluation scores were relatively close to each other, so no approach stood out. Among the two *TextRank* variations, the

implementation from *sumy* performed better on the datasets from which we extracted only one-sentence summaries (*XSum*, *XLSum and Reddit TIFU*), while the version from *pyTextRank* performed better on the rest of the chosen datasets. We also tested an embeddings-based extractive approach, *e-LexRank*, which in most datasets did not yield better results than the classical extractive approaches.

- Regarding our evaluation metrics, we note that the scores produced by applying *BLEU* are similar to those produced by *ROUGE*. This leads us to recommend the *BLEU* metric for the evaluation of summarization approaches, even if its original use concerned the field of machine translation.
- One may observe that the scores of **RL** match those of **RLSum** in all datasets, apart from the case of the *CNN/Daily Mail* dataset. With respect to the datasets that have single-sentence summaries, **RLSum** (summary-level) scores were equivalent to **RL** (sentence-level) scores. For the other datasets, which might contain multi-sentence summaries, this happens because the implementation of **RLSum** that we use splits sentences using the newline delimiter ($\backslash n$).
- The implication of the aforementioned experimental results is that all abstractive models do not perform equally, as also reported in [21]. Thus, there is a constant need for researchers to discover better pretrained language model architectures that generalize more easily and produce summaries closer to the human style of writing.

Our work also revealed a list of open issues in TS, which require further attention; these include:

- Abstractive models need to be retrained or refinetuned each time documents of different languages or different domains are introduced, respectively. This could be solved by creating more non-English datasets of various domains, and then training and finetuning different versions of the models.
- Instead of highly accurate but specialized abstractive models, a generic type of abstractive models could emerge. These could be trained and finetuned on vast multilingual corpora, achieving a degree of generalization that is not present in current models.
- Current abstractive approaches require a significant amount of training data [9,55] and training time, even with specialized hardware. This could be solved through the semisupervised nature of large language models (*LLMs*), e.g., *GPT-3* [52], which, due to their extremely large training and number of parameters (in the billion scale), can be finetuned for a specific language or domain through the utilization of a limited number of examples.
- As mentioned in Section 2.4, *BLEU* can be used as an evaluation metric for the TS task. As presented in Section 3.2.2, this metric produces a ranking for the approaches that is similar to that produced by *ROUGE*. To the best of our knowledge, most research works that evaluate summarization approaches utilize only the *ROUGE* metric.

Based on the above discussion and remarks, we propose the following list of future work directions:

- Evaluate *LLMs* for the TS task, given their zero/few-shot learning capabilities, which enable them to be finetuned for different languages or domains with a significantly smaller number of examples.
- Finetune existing abstractive approaches in other languages and/or domains.
- Utilize different evaluation metrics that do not penalize the approaches that produce abstractive summaries with synonymous terms (e.g., *BERTscore* [64], *BLEURT* [65], etc.).

Author Contributions: Conceptualization, N.G. and N.K.; methodology, N.G. and N.K.; software, N.G. and C.M.; investigation, N.G. and C.M.; resources, N.K.; data curation, C.M.; writing—original draft preparation, N.G. and C.M.; writing—review and editing, N.G., C.M., and N.K.; visualization, N.G. and C.M.; supervision, N.K.; project administration, N.K.; funding acquisition, N.K. All authors have read and agreed to the published version of the manuscript.

Funding: The work presented in this paper is supported by the inPOINT project (https://inpoint-project.eu/ (accessed on 4 April 2023)), which is cofinanced by the European Union and Greek national funds through the operational program, Competitiveness, Entrepreneurship, and Innovation, under the call RESEARCH—CREATE—INNOVATE (Project id: T2EDK-04389).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The employed datasets and the code repository are available at https://drive.google.com/drive/folders/1UJ_L5ZYYm52CQuURixc7DHs2h6Yz50F0?usp=sharing (accessed on 4 April 2023) and https://github.com/cmastrokostas/Automatic_Text_Summarization (accessed on 4 April 2023).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Sources of the publications selected in this review.

Journal/Conference/Workshop/Repository	Publisher
Nature	Nature
IEEE Access International Conference on Computer, Communication and Signal Processing (ICCCSP) IEEE Region 10 Symposium (TENSYMP)	Institute of Electrical and Electronics Engineers (IEEE)
Expert Systems with Applications Information Fusion Information Processing & Management Computer Speech & Language	Elsevier
AAAI Conference on Artificial Intelligence	Association for the Advancement of Artificial Intelligence (AAAI)
Journal of the American Society for Information Science and Technology (J. Am. Soc. Inf. Sci.)	Association for Computing Machinery (ACM)
Artificial Intelligence Review European Conference on Advances in Information Retrieval (ECIR) International Journal of Parallel Programming (Int J Parallel Prog)	Springer
International Conference on Language Resources and Evaluation (LREC) Conference on Empirical Methods in Natural Language Processing (EMNLP) Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) Annual Meeting of the Association for Computational Linguistics Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP) International Joint Conference on Natural Language Processing (IJCNLP) International Conference on Computational Linguistics: System Demonstrations (COLING) Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT) Findings of the Association for Computational Linguistics (ACL-IJCNLP) Text Summarization Branches Out Conference on Machine Translation (WMT) Workshop on New Frontiers in Summarization	Association for Computational Linguistics (ACL)
International Conference on Machine Learning (Proceedings of Machine Learning Research—PMLR) Journal of Machine Learning Research (JMLR)	JMLR, Inc. and Microtome Publishing (United States)
Journal of Artificial Intelligence Research (J. Artif. Int. Res.)	AI Access Foundation, Inc.

Table A1. *Cont.*

Journal/Conference/Workshop/Repository	Publisher
Journal of emerging technologies in web intelligence (JEWI)	JEWI
Advances in Neural Information Processing Systems	Curran Associates, Inc.
Foundations and Trends® in Information Retrieval	Now Publishers
IBM Journal of Research and Development	IBM
Arxiv preprints	Arxiv.org

References

1. Gupta, V.; Lehal, G.S. A Survey of Text Summarization Extractive Techniques. *J. Emerg. Technol. Web Intell.* **2010**, *2*, 258–268. [CrossRef]
2. El-Kassas, W.S.; Salama, C.R.; Rafea, A.A.; Mohamed, H.K. Automatic Text Summarization: A Comprehensive Survey. *Expert Syst. Appl.* **2021**, *165*, 113679. [CrossRef]
3. Bharti, S.K.; Babu, K.S. Automatic Keyword Extraction for Text Summarization: A Survey. *arXiv* **2017**, arXiv:1704.03242.
4. Gambhir, M.; Gupta, V. Recent Automatic Text Summarization Techniques: A Survey. *Artif. Intell. Rev.* **2017**, *47*, 1–66. [CrossRef]
5. Yasunaga, M.; Kasai, J.; Zhang, R.; Fabbri, A.R.; Li, I.; Friedman, D.; Radev, D.R. ScisummNet: A Large Annotated Corpus and Content-Impact Models for Scientific Paper Summarization with Citation Networks. *Proc. AAAI Conf. Artif. Intell.* **2019**, *33*, 7386–7393. [CrossRef]
6. An, C.; Zhong, M.; Chen, Y.; Wang, D.; Qiu, X.; Huang, X. Enhancing Scientific Papers Summarization with Citation Graph. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 12498–12506. [CrossRef]
7. Hong, K.; Conroy, J.; Favre, B.; Kulesza, A.; Lin, H.; Nenkova, A. A Repository of State of the Art and Competitive Baseline Summaries for Generic News Summarization. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland, 26–31 May 2014; European Language Resources Association (ELRA): Luxembourg; pp. 1608–1616.
8. Narayan, S.; Cohen, S.B.; Lapata, M. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 1797–1807.
9. Zhang, J.; Zhao, Y.; Saleh, M.; Liu, P. PEGASUS: Pre-Training with Extracted Gap-Sentences for Abstractive Summarization. In Proceedings of the 37th International Conference on Machine Learning, Virtual Event. 21 November 2020; PMLR. pp. 11328–11339.
10. Zhang, S.; Celikyilmaz, A.; Gao, J.; Bansal, M. EmailSum: Abstractive Email Thread Summarization. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online. 1–6 August 2021; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 6895–6909.
11. Polsley, S.; Jhunjhunwala, P.; Huang, R. CaseSummarizer: A System for Automated Summarization of Legal Texts. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, Osaka, Japan, 11–16 December 2016; The COLING 2016 Organizing Committee. pp. 258–262.
12. Kanapala, A.; Pal, S.; Pamula, R. Text Summarization from Legal Documents: A Survey. *Artif. Intell. Rev.* **2019**, *51*, 371–402. [CrossRef]
13. Bhattacharya, P.; Hiware, K.; Rajgaria, S.; Pochhi, N.; Ghosh, K.; Ghosh, S. A Comparative Study of Summarization Algorithms Applied to Legal Case Judgments. In *Advances in Information Retrieval, Proceedings of the 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, 14–18 April 2019*; Azzopardi, L., Stein, B., Fuhr, N., Mayr, P., Hauff, C., Hiemstra, D., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 413–428.
14. Sun, S.; Luo, C.; Chen, J. A Review of Natural Language Processing Techniques for Opinion Mining Systems. *Inf. Fusion* **2017**, *36*, 10–25. [CrossRef]
15. Hu, Y.H.; Chen, Y.L.; Chou, H.L. Opinion Mining from Online Hotel Reviews—A Text Summarization Approach. *Inf. Process. Manag.* **2017**, *53*, 436–449. [CrossRef]
16. Adamides, E.; Giarelis, N.; Kanakaris, N.; Karacapilidis, N.; Konstantinopoulos, K.; Siachos, I. Leveraging open innovation practices through a novel ICT platform. In *Human Centred Intelligent Systems, Proceedings of KES HCIS 2023 Conference. Smart Innovation, Systems and Technologies, Rome, Italy, 14–16 June 2023*; Springer: Rome, Italy, 2023; Volume 359.
17. Nenkova, A.; McKeown, K. Automatic Summarization. *Found. Trends Inf. Retr.* **2011**, *5*, 103–233. [CrossRef]
18. Saggion, H.; Poibeau, T. Automatic Text Summarization: Past, Present and Future. In *Multi-Source, Multilingual Information Extraction and Summarization*; Poibeau, T., Saggion, H., Piskorski, J., Yangarber, R., Eds.; Theory and Applications of Natural Language Processing; Springer: Berlin/Heidelberg, Germany, 2013; pp. 3–21. ISBN 9783642285691.

19. Moratanch, N.; Chitrakala, S. A Survey on Extractive Text Summarization. In Proceedings of the 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP), Chennai, India, 10–11 January 2017; pp. 1–6.
20. Mridha, M.F.; Lima, A.A.; Nur, K.; Das, S.C.; Hasan, M.; Kabir, M.M. A Survey of Automatic Text Summarization: Progress, Process and Challenges. *IEEE Access* **2021**, *9*, 156043–156070. [CrossRef]
21. Alomari, A.; Idris, N.; Sabri, A.Q.M.; Alsmadi, I. Deep Reinforcement and Transfer Learning for Abstractive Text Summarization: A Review. *Comput. Speech Lang.* **2022**, *71*, 101276. [CrossRef]
22. Lin, C.Y. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*; Association for Computational Linguistics: Barcelona, Spain, 2004; pp. 74–81.
23. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics; Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002; pp. 311–318.
24. Graham, Y. Re-Evaluating Automatic Summarization with BLEU and 192 Shades of ROUGE. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 128–137.
25. Rieger, B.B. *On Distributed Representation in Word Semantics*; International Computer Science Institute: Berkeley, CA, USA, 1991.
26. Luhn, H.P. The Automatic Creation of Literature Abstracts. *IBM J. Res. Dev.* **1958**, *2*, 159–165. [CrossRef]
27. Deerwester, S.; Dumais, S.T.; Furnas, G.W.; Landauer, T.K.; Harshman, R. Indexing by Latent Semantic Analysis. *J. Am. Soc. Inf. Sci.* **1990**, *41*, 391–407. Available online: https://search.crossref.org/?q=Indexing+by+latent+semantic+analysis+Scott+Deerwester&from_ui=yes (accessed on 30 May 2023). [CrossRef]
28. Gong, Y.; Liu, X. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. In Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, 1 September 2001; Association for Computing Machinery: New York, NY, USA; pp. 19–25.
29. Steinberger, J.; Jezek, K. Using Latent Semantic Analysis in Text Summarization and Summary Evaluation. *Proc. ISIM* **2004**, *4*, 8.
30. Yeh, J.Y.; Ke, H.R.; Yang, W.P.; Meng, I.H. Text Summarization Using a Trainable Summarizer and Latent Semantic Analysis. *Inf. Process. Manag.* **2005**, *41*, 75–95. [CrossRef]
31. Mihalcea, R.; Tarau, P. TextRank: Bringing Order into Text. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 25–26 July 2004; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 404–411.
32. Page, L.; Brin, S.; Motwani, R.; Winograd, T. *The Pagerank Citation Ranking: Bring Order to the Web*; Technical Report; Stanford University: Stanford, CA, USA, 1998.
33. Erkan, G.; Radev, D.R. LexRank: Graph-Based Lexical Centrality as Salience in Text Summarization. *J. Artif. Int. Res.* **2004**, *22*, 457–479. [CrossRef]
34. Bougouin, A.; Boudin, F.; Daille, B. TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction. In Proceedings of the Sixth International Joint Conference on Natural Language Processing, Nagoya, Japan, 14–19 October 2013; Asian Federation of Natural Language Processing: Singapore; pp. 543–551.
35. Florescu, C.; Caragea, C. PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, CMA, Canada, 30 July–4 August 2017; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 1105–1115.
36. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
37. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 3982–3992.
38. Chengzhang, X.; Dan, L. Chinese Text Summarization Algorithm Based on Word2vec. *J. Phys. Conf. Ser.* **2018**, *976*, 012006. [CrossRef]
39. Haider, M.M.; Hossin, M.d.A.; Mahi, H.R.; Arif, H. Automatic Text Summarization Using Gensim Word2Vec and K-Means Clustering Algorithm. In Proceedings of the 2020 IEEE Region 10 Symposium (TENSYMP), Dhaka, Bangladesh, 5–7 June 2020; pp. 283–286.
40. Abdulateef, S.; Khan, N.A.; Chen, B.; Shang, X. Multidocument Arabic Text Summarization Based on Clustering and Word2Vec to Reduce Redundancy. *Information* **2020**, *11*, 59. [CrossRef]
41. Ganesan, K.; Zhai, C.; Han, J. Opinosis: A Graph Based Approach to Abstractive Summarization of Highly Redundant Opinions. In Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), Beijing, China, 23–27 August 2010; Coling 2010 Organizing Committee. pp. 340–348.
42. Genest, P.E.; Lapalme, G. Fully Abstractive Approach to Guided Summarization. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Jeju Island, Korea, 8–14 July 2012; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 354–358.

43. Khan, A.; Salim, N.; Farman, H.; Khan, M.; Jan, B.; Ahmad, A.; Ahmed, I.; Paul, A. Abstractive Text Summarization Based on Improved Semantic Graph Approach. *Int. J. Parallel. Prog.* **2018**, *46*, 992–1016. [CrossRef]
44. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
45. Rekabdar, B.; Mousas, C.; Gupta, B. Generative Adversarial Network with Policy Gradient for Text Summarization. In Proceedings of the 2019 IEEE 13th International Conference on Semantic Computing (ICSC), Newport Beach, CA, USA, 30 January–1 February 2019; pp. 204–207.
46. Yang, M.; Li, C.; Shen, Y.; Wu, Q.; Zhao, Z.; Chen, X. Hierarchical Human-Like Deep Neural Networks for Abstractive Text Summarization. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 2744–2757. [CrossRef] [PubMed]
47. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
48. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2016**, arXiv:1409.0473.
49. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.
50. Xue, L.; Constant, N.; Roberts, A.; Kale, M.; Al-Rfou, R.; Siddhant, A.; Barua, A.; Raffel, C. MT5: A Massively Multilingual Pre-Trained Text-to-Text Transformer. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online. 8 June 2021; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 483–498.
51. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-Training for Natural Language Generation, Translation, and Comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online. 10 July 2020; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 7871–7880.
52. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models Are Few-Shot Learners. In Proceedings of the Advances in Neural Information Processing Systems, Online. 6–12 December 2020; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 1877–1901.
53. Shleifer, S.; Rush, A.M. Pre-Trained Summarization Distillation. *arXiv* **2020**, arXiv:2010.13002.
54. Hermann, K.M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; Blunsom, P. Teaching Machines to Read and Comprehend. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, CMA, Canada, 7–12 December 2015; Curran Associates, Inc.: Red Hook, NY, USA, 2015; Volume 28.
55. Gliwa, B.; Mochol, I.; Biesek, M.; Wawer, A. SAMSum Corpus: A Human-Annotated Dialogue Dataset for Abstractive Summarization. In Proceedings of the 2nd Workshop on New Frontiers in Summarization, Hong Kong, China, 4 November 2019; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 70–79.
56. Kim, B.; Kim, H.; Kim, G. Abstractive Summarization of Reddit Posts with Multi-Level Memory Networks. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, MN, USA, 2–7 June 2019; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 2519–2531.
57. Kornilova, A.; Eidelman, V. BillSum: A Corpus for Automatic Summarization of US Legislation. In Proceedings of the 2nd Workshop on New Frontiers in Summarization, Hong Kong, China, 4 November 2019; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 48–56.
58. Hasan, T.; Bhattacharjee, A.; Islam, M.d.S.; Mubasshir, K.; Li, Y.F.; Kang, Y.B.; Rahman, M.S.; Shahriyar, R. XL-Sum: Large-Scale Multilingual Abstractive Summarization for 44 Languages. In Proceedings of the Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, Online, 1–6 August 2021; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 4693–4703.
59. Koh, H.Y.; Ju, J.; Liu, M.; Pan, S. An Empirical Survey on Long Document Summarization: Datasets, Models, and Metrics. *ACM Comput. Surv.* **2022**, *55*, 1–35. [CrossRef]
60. Post, M. A Call for Clarity in Reporting BLEU Scores. In Proceedings of the Third Conference on Machine Translation: Research Papers, Brussels, Belgium, 31 October–1 November 2018; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 186–191.
61. Nathan, P. PyTextRank, a Python Implementation of TextRank for Phrase Extraction and Summarization of Text Documents. DerwenAI/Pytextrank: v3.1.1 release on PyPi | Zenodo. 2016. Available online: https://zenodo.org/record/4637885 (accessed on 19 June 2023).
62. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. Transformers: State-of-the-Art Natural Language Processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online. 5 October 2020; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 38–45.
63. Fabbri, A.; Li, I.; She, T.; Li, S.; Radev, D. Multi-News: A Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 1074–1084.

64. Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K.Q.; Artzi, Y. BERTScore: Evaluating Text Generation with BERT. *arXiv* **2020**, arXiv:1904.09675.
65. Sellam, T.; Das, D.; Parikh, A. BLEURT: Learning Robust Metrics for Text Generation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 7881–7892.