**NEWS ARTICLES SUMMARISATION**

Bhanu Sama, Prashanth Mourya, Sai Kumar Gheereddy, Sumedha Sirikonda

University of Maryland Baltimore County

1000 Hilltop Circle, Baltimore, MD 21250

(December 20, 2021)

**ABSTRACT**

In recent years, the volume of textual data has rapidly increased, which has generated a valuable resource for extracting and analyzing information. To retrieve useful knowledge within a reasonable time, this information must be summarized. In this paper, we present NEWSROOM, a summarization dataset of 1.3 million articles and summaries written by authors and editors in newsrooms of 38 major news publications. Extracted from search and social media metadata between 1998 and 2017, these high-quality summaries demonstrate high diversity of summarization styles. In particular, the summaries are abstractive strategies, borrowing words and phrases from articles at varying rates. We analyze the NEWSROOM articles and summaries and provide an abstractive summary of the article by applying transfer learning on the existing models.

**I. INTRODUCTION**

In the modern era of big data, there are vast quantities of textual data available, including online documents, articles, news, and reviews that contain long strings of text that need to be summarized [1]. The importance of text summarization is due to several reasons, including the retrieval of significant information from a long text within a short period, easy and rapid loading of the most important information, and resolution of the problems associated with the criteria needed for summary evaluation [2]**.** The task of the text summarization is to condense long documents into short summaries while preserving the important information and meaning of the documents. According to Radef [3] a summary is defined as “a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually, significantly less than that”. Automatic text summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning. In recent years, numerous approaches have been developed for automatic text summarization and applied widely in various domains.

In this paper, we present NEWSROOM, a dataset with 1.3 million news articles and human-written summaries. NEWSROOM’s summaries were written by authors and editors in the newsrooms of news, sports, entertainment, financial, and other publications. NEWSROOM summaries are written by humans, for common readers, and with the explicit purpose of summarization. As a result, NEWSROOM is a nearly two decade-long snapshot representing how single document summarization is used in practice across a variety of sources, writers, and topics. Identifying large, high-quality resources for summarization has called for creative solutions in the past. This includes using news headlines as summaries of article prefixes[4][5], concatenating bullet points as summaries[6], or using librarian archival summaries[7]. While these solutions provide large scale data, it comes at the cost of how well they reflect the summarization problem or their focus on very specific styles of summarizations. NEWSROOM is distinguished from these resources in its combination of size and diversity. The summaries were written with the explicit goal of concisely summarizing news articles over almost two decades. Rather than rely on a single source, the dataset includes summaries from 38 major publishers. This diversity of sources and time span translate into a diversity of summarization styles.

We explore NEWSROOM to better understand the dataset and apply Transfer learning summarization techniques on the news articles. We focus on a key dimension, abstractive summarization : abstractive summarization rephrases the original text to generate new phrases that may not be in the original text, which is considered a difficult task for a computer. As abstractive text summarization requires an understanding of the document to generate the summary, advanced machine learning techniques and extensive natural language processing (NLP) are required. Thus, abstractive summarization is harder than extractive summarization since abstractive summarization requires real-word knowledge and semantic class analysis [8]. However, abstractive summarization is also better than extractive summarization since the summary is an approximate representation of a human-generated summary, which makes it more meaningful [9]. We have used pre-trained models T5 transformer[10] and BERT[11][12] to generate abstractive summaries for the news articles in NEWSROOM dataset.

**II. LITERATURE REVIEW**

Deep learning techniques were employed in abstractive text summarisation for the first time in 2015 [13], and the proposed model was based on the encoder-decoder architecture. For these applications, deep learning techniques have provided excellent results and have been extensively employed in recent years.

Raphal et al. surveyed several abstractive text summarisation processes in general [14]. Their study differentiated between different model architectures, such as reinforcement learning (RL), supervised learning, and attention mechanism. In addition, comparisons in terms of word embedding, data processing, training, and validation had been performed. However, there are no comparisons of the quality of several models that generated summaries. Furthermore, both extractive and abstractive summarization models were summarized in [15, 16]. In [15], the classification of summarization tasks was based on three factors: input factors, purpose factors, and output factors. Dong and Mahajani et al. surveyed only five abstractive summarization models each. On the other hand, Mahajani et al. focused on the datasets and training techniques in addition to the architecture of several abstractive summarization models [16]. However, the quality of the generated summary of the different techniques and the evaluation measures were not discussed.

Shi et al. presented a comprehensive survey of several abstractive text summarization models, which are based on sequence-to-sequence encoder-decoder architecture for convolutional and RNN seq2seq models. The focus was the structure of the network, training strategy, and the algorithms employed to generate the summary [17]. Although several papers have analyzed abstractive summarization models, few papers have performed a comprehensive study [18]. Deep learning analyses complex problems to facilitate the decision-making process. Deep learning attempts to imitate what the human brain can achieve by extracting features at different levels of abstraction. Typically, higher-level layers have fewer details than lower-level layers [19]. The output layer will produce an output by nonlinearly transforming the input from the input layer. The hierarchical structure of deep learning can support learning. The level of abstraction of a certain layer will determine the level of abstraction of the next layer since the output of one layer will be the input of the next layer. In addition, the number of layers determines the deepness, which affects the level of learning [20].

Deep learning is applied in several NLP tasks since it facilitates the learning of multilevel hierarchal representations of data using several data processing layers of nonlinear units [19, 21–23]. Various deep learning models have been employed for abstractive summarisation, including RNNs, convolutional neural networks (CNNs), and sequence-to-sequence models. For our project we have used T5 and BERT Transformer deep learning models.

**III. DATASET**

We have retrieved the dataset from the NEWSROOM website which is publicly available. Newsroom contains two scripts for downloading and processing data downloaded from Archive.org. First, download the "Thin Dataset" from <https://summari.es/download/>.

Timeline

Description automatically generated with medium confidence

Fig1: The above figure shows how data is obtained

Download and extract thin.tar with tar xvf thin.tar, yielding directory thin containing several .jsonl.gz files. Next, we used newsroom scrape and newsroom extract to process the data, then we have stored the data in Mongo DB and using pymongo we read the data as a dataframe.

**Data Pre-Processing**

In our Project, since we are dealing with textual data it is important to preprocess the data. The data was cleaned, converted into a dataframe to analyze the patterns (visible and missing).

Table

Description automatically generated

Fig2: The above figure shows data information

Corpus preparation and cleaning were done using a series of packages running on top of Python such as the Natural Language Toolkit (NLTK) that provides stop-word removal, stemming, lemmatizing, tokenization, identifying n-gram procedures, and other data cleanings like lowercase transformation and punctuation removal. The preprocessing steps are supported in NLTK Library and contain the following patterns:

• Stop-word elimination: Removal of the most common words in a language that are not helpful and in general unusable in text mining like prepositions, numbers, and words that do not contain applicable information for the study. In fact, in NLP, there is no general list of stop words used by all developers who choose their list based on their goal to improve the recommendation system performance.

• Stemming: Convert words into their root, using stemming algorithms like Snowball Stemmer.

• Lemmatizing: Enhances system accuracy by returning the base or dictionary form of a word.

• Tokenizing: Divide the text input to tokens like phrases, words, or other meaningful elements resulting in a sequence of tokens.

**IV. Data Analysis**

NEWSROOM is a large dataset for training and evaluating summarization systems. It contains 1.3 million articles and summaries written by authors and editors in the newsrooms of 38 major publications. It contains summaries from different topic domains, written by many authors, over the span of more than two decades. This diversity is an important aspect of the dataset. This is how a single object in the data looks like.

Graphical user interface, text, application

Description automatically generated

Fig3: The above figure shows data representation

Dataset features includes: - text: Input news text. - summary: Summary for the news. And additional features: - title: news title. url: url of the news. - date: date of the article. - density: extractive density. coverage: extractive coverage. compression: compression ratio. density\_bin: low, medium, high. coverage\_bin: extractive, abstractive. compression\_bin: low, medium, high.

**Characterizing Summarization Strategies** We examine summarization strategies using three measures that capture the degree of text overlap between the summary and article, and the rate of compression of the information conveyed. Given an article text A = ha1, a2, . . . , ani consisting of a sequence of tokens ai and the corresponding article summary S = hs1, s2, · · · , smi consisting of tokens si , the set of extractive fragments F(A, S)is the set of shared sequences of tokens in A and S. We identify these extractive fragments of an article-summary pair using a greedy process. We process the tokens in the summary in order. At each position, if there is a sequence of tokens in the source text that is prefix of the remainder of the summary, we mark this prefix as extractive and continue. We prefer to mark the longest prefix possible at each step. Otherwise, we mark the current summary token as abstractive. The set F(A, S) includes all the tokens sequences identified as extractive. Figure 3 formally describes this procedure. Underlined phrases of Figures 1 and 2 are examples of fragments identified as extractive. Using F(A, S), we compute two measures: extractive fragment coverage and extractive fragment density[24].

**Extractive Fragment Coverage** The coverage measure quantifies the extent to which a summary is derivative of a text. COVERAGE(A, S) measures the percentage of words in the summary that are part of an extractive fragment with the article: COVERAGE(A, S) = 1 |S| X f∈F(A,S) |f| . For example, a summary with 10 words that borrows 7 words from its article text and includes 3 new words will have COVERAGE(A, S) = 0.7.

**Extractive Fragment Density** The density measure quantifies how well the word sequence of a summary can be described as a series of extractions. For instance, a summary might contain many individual words from the article and therefore have a high coverage. However, if arranged in a new order, the words of the summary could still be used to convey ideas not present in the article. We define DENSITY(A, S) as the average length of the extractive fragment to which each word in the summary belongs. The density formulation is similar to the coverage definition but uses a square of the fragment length: DENSITY(A, S) = 1 |S| X f∈F(A,S) |f| 2 . For example, an article with a 10-word summary made of two extractive fragments of lengths 3 and 4 would have COVERAGE(A, S) = 0.7 and DENSITY(A, S) = 2.5[24].

**Compression Ratio** We use a simple dimension of summarization, compression ratio, to further characterize summarization strategies. We define COMPRESSION as the word ratio between the article and summary: COMPRESSION(A, S) = |A|/|S| . Summarizing with higher compression is challenging as it requires capturing more precisely the critical aspects of the article text[24].

**Topic Modelling**

Topic modeling is a machine learning technique that automatically analyzes text data to determine cluster words for a set of documents. This is known as ‘unsupervised’ machine learning because it does not require a predefined list of tags or training data that has been previously classified by humans. Topic modeling involves counting words and grouping similar word patterns to infer topics within unstructured data. We did topic modelling to analyze the topics in the articles. We have used LDA model for topic modelling.

**Latent Dirichlet Allocation(LDA)**: Latent Dirichlet Allocation (LDA) is based on the same underlying assumptions: the distributional hypothesis, (i.e., similar topics make use of similar words) and the statistical mixture hypothesis (i.e., documents talk about several topics) for which a statistical distribution can be determined. The purpose of LDA is to map each document in our corpus to a set of topics that covers a good deal of the words in the article. LDA assumes that the distribution of topics in a article and the distribution of words in topics are Dirichlet distributions. Two hyperparameters control article and topic similarity, known as alpha and beta, respectively. A low value of alpha will assign fewer topics to each document whereas a high value of alpha will have the opposite effect. A low value of beta will use fewer words to model a topic whereas a high value will use more words, thus making topics more similar between them. A third hyperparameter must be set when implementing LDA, namely, the number of topics the algorithm will detect since LDA cannot decide on the number of topics by itself [26]. In our project, we have used the Genism library for LDA. In Genism, a document is an object of the text sequence type (string). First, we obtain an id-2-word dictionary. For each headline, we will use the dictionary to obtain a mapping of the word id to their word counts. The LDA model uses both mappings. Next by generating LDA topics We will iterate over the number of topics, get the top words in each cluster, and add them to a data frame. Below are some of the results from topic modelling

Text

Description automatically generated

Fig 4: The above figure shows output of topic modelling

**V. METHODS**

We have used the start of the art models T5 and BERT to generate summaries for the articles in the dataset.

**BERT**

Bidirectional Encoder Representations from Transformers is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering, Natural Language Inference, and others. BERT’s key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This contrasts with previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. We introduce BERT and its detailed implementation in this section. There are two steps in our framework: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For finetuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters. The question-answering example in Figure 1 will serve as a running example for this section. A distinctive feature of BERT is its unified architecture across different tasks. There is minimal difference between the pre-trained architecture and the final downstream architecture[29].

A screenshot of a computer

Description automatically generated with medium confidence

Fig 5: The above figure shows BERT architecture

The paper’s results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible. BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT’s goal is to generate a language model, only the encoder mechanism is necessary. The detailed workings of Transformer are described in a paper by Google. Here we are adding a pictorial representation of how our model works.

**T5**

The Text-to-Text Transfer Transformer or T5 is a type of Transformer that is capable of being trained on a variety of tasks with a uniform architecture. It was created by Google AI and was published about in the paper Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer[25]. Raffel et al[25] present a large-scale empirical survey to determine which transfer learning techniques work best and apply these insights at scale to create a new model called the Text-To-Text Transfer Transformer (T5). They propose reframing all NLP tasks into a unified text-to-text-format where the input and output are always text strings.

Graphical user interface, text, application

Description automatically generated

Fig 6: The above figure shows Text-to-text framework

T5 is an encoder-decoder model and converts all NLP problems into a text-to-text format. It is trained using teacher forcing. This means that for training, we always need an input sequence and a corresponding target sequence. The input sequence is fed to the model using input ids. The target sequence is shifted to the right, i.e., prepended by a start-sequence token and fed to the decoder using the decoder input ids. In teacher-forcing style, the target sequence is then appended by the EOS token and corresponds to the labels. The PAD token is hereby used as the start-sequence token. T5 can be trained / fine-tuned both in a supervised and unsupervised fashion.

We make use of the pipeline function from the huggingface api [27] to run the Text Summarization process using the T-5 Model. We pass a thousand records of our data from our dataset into the pipeline function to generate the summary. We make use of the rouge metric for evaluation of our model. We use the rouge-1, rouge-2, and rouge-l evaluation techniques.

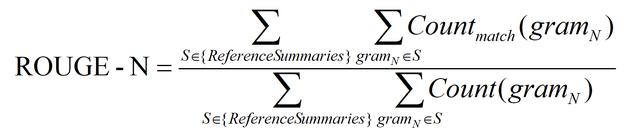
Graphical user interface, text

Description automatically generated

Fig 7: The above figure shows Code for generating T5 scores

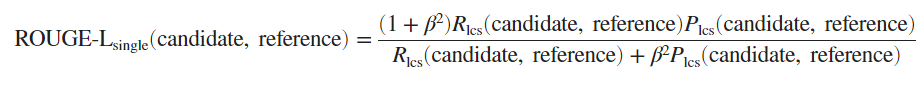
**VI. EVALUATION METRICS**

We use the rouge metric introduced by Lin[28] to evaluate results. The package ROUGE is employed to evaluate the text summarisation techniques by comparing the generated summary with a manually generated summary. The package consists of several measures to evaluate the performance of text summarisation techniques, such as ROUGE-N (ROUGE1 and ROUGE2) and ROUGE-L, which were employed in several studies [30]v. ROUGE-N is *n*-gram recall such that ROUGE1 and ROUGE2 are related to unigrams and bigrams, respectively, while ROUGE-L is related to the longest common substring. Since the manual evaluation of automatic text summarization is a time-consuming process and requires extensive effort, ROUGE is employed as a standard for evaluating text summarization. ROUGE-N is calculated using the following equation:



 where *S* is the reference summary, *n* is the *n*-gram length, and Countmatch (gram*n*) is the maximum number of matching *n*-gram words between the reference summary and the generated summary. Count (gram*n*) is the total number of *n*-gram words in the reference summary [31].

We also use the Rouge-L score where L stands for Longest Common Subsequence. The formula is given by



where the parameter *β* controls the relative importance of the precision and recall. Because the ROUGE score favors recall, *β* is typically set to a high value.

Each rouge score has a ‘f’, ‘p’ and ‘r’ metric which stands for F1-Score, precision, and recall respectively.

Chart, bar chart

Description automatically generated

Fig 8: Score Comparison of Bert and T5

As we can see from the graph above, the Bert Model, given in blue, consistently outperforms in and generates more consistent summaries as compared to the T5 model. Given in blue.

**Chart, scatter chart

Description automatically generated**

Fig 9: Rouge-1 score of every article

The graph above shows us the scatter plot of the rouge-1 F1 score of all the data points for the Bert (given in blue) and T5 Model (given in orange). The Bart model has a higher average score compared to the T5 Model.

This is how our results for Rouge metrics looks like

**Table

Description automatically generated**

**VII. RESULTS**

The T5 transformer model summarizes the news articles and output is shown below.

**Text

Description automatically generated**

The BERT model summarizes the news articles and output is shown below **Graphical user interface, text, application, email

Description automatically generated**

**VIII. CONCLUSION**

In this paper, we present NEWSROOM, a dataset of articles and their summaries written in the newsrooms of online publications. NEWSROOM is the largest summarization dataset available to date and exhibits a wide variety of human summarization strategies. we have explained the Bert and T5 summarization techniques used in our project news article summarization. We used the newsroom script to process the data and did data pre-processing and analysis. also, we did topic modelling to find the most popular topics in the news articles. Next, we have implemented transfer learning for the pre-trained Bert and T5 models to generate abstractive summaries for the news articles. After running the dataset through our models, we found out that the T5 Model, having an average F1 score of about 33% outshines the Bert model that has an average F1 score of about 26%. It also has better metrics when it comes to recall and precision across all rouge-1, rouge-2, and rouge-l scores. Hence, we can say conclusively that the T5 model was more efficient in summarizing news articles as compared to the Bert model for our dataset.

**Source Code**

The source code for this project can be found on Github at: https://github.com/Sumedha-Sirikonda/News-Article-Summarization

**BIBLIOGRAPHY**

1. M. Allahyari, S. Pouriyeh, M. Assefi et al., “Text summarization techniques: a brief survey,” *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 10, 2017.

2. A. B. Al-Saleh and M. E. B. Menai, “Automatic Arabic text summarization: a survey,” *Artificial Intelligence Review*, vol. 45, no. 2, pp. 203–234, 2016.

3.Dragomir R Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. Computational linguistics 28, 4 (2002), 399–408.

4. Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction. Association for Computational Linguistics, Stroudsburg, PA, USA, AKBC-WEKEX ’12, pages 95–100. https://www.cs.cmu.edu/~mgormley/papers/napoles+gormley+van-durme.naaclw.2012.pdf

5. Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015. pages 379–389.

6.Karl Moritz Hermann, Toma´s Ko ˇ cisk ˇ y, Edward ´ Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1. MIT Press, Cambridge, MA, USA, NIPS’15, pages 1693– 1701. http://dl.acm.org/citation.cfm?id=2969239. 2969428.

7. E. Sandhaus. 2008. The New York Times Annotated Corpus. Linguistic Data Consortium, Philadelphia 6(12).

8. Q. A. Al-Radaideh and D. Q. Bataineh, “A hybrid approach for Arabic text summarization using domain knowledge and genetic algorithms,” *Cognitive Computation*, vol. 10, no. 4, pp. 651–669, 2018.

9. C. Sunitha, A. Jaya, and A. Ganesh, “A study on abstractive summarization techniques in Indian languages,” *Procedia Computer Science*, vol. 87, pp. 25–31, 2016.

10.https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html

11.https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/

12.https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

13. A. M. Rush, S. Chopra, and J. Weston, “A neural attention model for abstractive sentence summarization,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

14. N. Raphal, H. Duwarah, and P. Daniel, “Survey on abstractive text summarization,” in *Proceedings of the 2018 International Conference on Communication and Signal Processing (ICCSP)*, pp. 513–517, Chennai, 2018.

15. Y. Dong, “A survey on neural network-based summarization methods,” 2018, <http://arxiv.org/abs/1804.04589>.

16.A. Mahajani, V. Pandya, I. Maria, and D. Sharma, “A comprehensive survey on extractive and abstractive techniques for text summarization,” in *Ambient Communications and Computer Systems*, Y.-C. Hu, S. Tiwari, K. K. Mishra, and M. C. Trivedi, Eds., vol. 904, pp. 339–351, Springer, Singapore, 2019.

17.T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, *Neural Abstractive Text Summarization with Sequence-To-Sequence Models: A Survey*, 2020, <http://arxiv.org/abs/1812.02303>.

18.A. Joshi, E. Fidalgo, E. Alegre, and U. de León, “Deep learning based text summarization: approaches, databases and evaluation measures,” in *Proceedings of the International Conference of Applications of Intelligent Systems*, Spain, 2018.

19.Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

20.D. Suleiman, A. Awajan, and W. Al Etaiwi, “The use of hidden Markov model in natural Arabic language processing: a survey,” *Procedia Computer Science*, vol. 113, pp. 240–247, 2017.

21.H. Wang and D. Zeng, “Fusing logical relationship information of text in neural network for text classification,” *Mathematical Problems in Engineering*, vol. 2020, pp. 1–16, 2020.

22.J. Yi, Y. Zhang, X. Zhao, and J. Wan, “A novel text clustering approach using deep-learning vocabulary network,” *Mathematical Problems in Engineering*, vol. 2017, pp. 1–13, 2017.

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

24. Max Grusky, Mor Naaman, Yoav Artzi NEWSROOM: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies.

25.Colin Raffel and Noam Shazeer and Adam Roberts and Katherine Lee and Sharan Narang and Michael Matena and Yanqi Zhou and Wei Li and Peter J. Liu (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. CoRR, abs/1910.10683.

26.Federico Pascual Topic Modeling: An Introduction (2019).

27. Thomas Wolf and Lysandre Debut and Victor Sanh and Julien Chaumond and Clement Delangue and Anthony Moi and Pierric Cistac and Tim Rault and Rémi Louf and Morgan Funtowicz and Jamie Brew (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing. CoRR, abs/1910.03771.

28.Lin, C. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. ACL 2004.

29. Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

30. E. Jobson and A. Gutiérrez, *Abstractive Text Summarization Using Attentive Sequence-To-Sequence RNNs*, p. 8, 2016.

31. C.-Y. Lin, “ROUGE: a package for automatic evaluation of summaries,” in *Proceedings of the 2004 ACL Workshop*, Barcelona, Spain, July 2004.