



VIT University

Research paper title generation

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Computer Science Technology
by

Prakrit Timilsina (20BCE2764)
Anish Shrestha (20BCE2893)
Aviral Sharma (20BCE2918)
Md Anwar Mansuri(20BCE2892)

Team ID: 21
E1 Slot

CSE4022 –Natural Language Processing
Under the guidance of
Rajeshkannan R

Abstract

GPT-2 research paper title generator is a novel approach to automatically generating research paper titles. Open-GPT-2 AI's model is a cutting-edge language generation model that has been trained on massive amounts of text data. This enables it to generate text that is highly coherent and fluent on a wide range of topics. In this project, we propose using GPT-2 to generate titles for research papers in fields ranging from computer science to biology to economics. The primary goal of using GPT-2 for research paper title generation is to take advantage of the model's ability to comprehend the context and content of a given text. GPT-2 can generate a title that accurately reflects the main ideas and contributions of a research paper if it is given a brief summary or outline of the paper. This can save researchers a lot of time and effort when it comes to coming up with a good title for their paper. To test the efficacy of our proposed method, we will run experiments on a dataset of research papers from various fields. The generated titles will be compared to the actual paper titles and evaluated using metrics such as relevance, fluency, and grammatical correctness. In addition, we will conduct a human evaluation to determine the overall quality of the generated titles. In summary, the purpose of this study is to investigate the potential of GPT-2 for research paper title generation and assess its efficacy in terms of relevance, fluency, and grammatical correctness. The proposed method has the potential to save researchers significant time and effort in developing an appropriate title for their papers.

Introduction

The ability to generate accurate and informative titles for PDF files is essential in various fields. For example, in research, generating a concise and informative title can make a significant difference in the visibility and impact of a study. In publishing, titles can greatly influence the success of a publication by capturing the attention of potential readers. And in education, generating accurate titles for course materials can help students navigate and comprehend the content better.

However, manual title generation can be time-consuming, and errors can occur due to factors such as subjectivity, cognitive bias, and lack of expertise. The web application developed in this project addresses these challenges by providing an automated solution that leverages advanced text generation and analysis techniques.

The Flask framework used in this project is a popular choice for developing web applications due to its flexibility, scalability, and ease of use. It allows for the creation of a powerful and intuitive user interface that simplifies the title generation process, enabling users to upload PDF files and enter actual titles with ease. The OpenAI text generation API is a cutting-edge tool that uses advanced natural language processing techniques to generate human-like text based on a given prompt. This ensures that the generated titles are relevant and informative, and can save users significant time and effort compared to manual generation. The SequenceMatcher library used in the project is a powerful tool for calculating the similarity between two strings, providing an accurate measure of the accuracy of the generated title compared to the actual title. The use of a JSON file to store the accuracy data allows for easy retrieval and analysis, enabling users to identify patterns and trends in their title generation data.

Finally, the use of the PyPDF2 library to extract text from PDF files ensures that the generated titles are based on the content of the file, making them more relevant and informative. And the Matplotlib library used to create a line chart of the accuracy values provides a useful visualisation tool for analysing the title generation data.

In conclusion, the web application developed in this project provides a valuable tool for generating accurate and informative titles for PDF files in various fields. With its user-friendly interface, advanced text generation and analysis techniques, and powerful visualisation capabilities, this application is sure to be a valuable asset to many users.

GPT-2

GPT-2 is a state-of-the-art language model that uses a neural network architecture called the transformer to generate human-like text. It was developed by OpenAI and was released in 2019. GPT-2 is pre-trained on a large corpus of text data and is capable of generating high-quality text in a variety of styles and topics. It has 1.5 billion parameters, making it one of the largest and most powerful language models available.

Our code uses a supervised learning algorithm to train a GPT-2 model for generating titles from abstracts. The algorithm involves several steps, including data preprocessing, defining a neural network architecture, setting up a training loop, and optimising the model parameters using a stochastic gradient descent algorithm.

At a high level, the algorithm works by feeding batches of input data (in this case, abstracts) to the model, which generates corresponding output data (in this case, titles). The model is then optimised using a loss function that measures the difference between the predicted output and the actual output. This process is repeated for multiple epochs until the model converges to a set of parameters that minimise the loss function.

GPT-2 is a highly parallelizable model that can be trained and run on multiple devices simultaneously. This is because the transformer architecture used in GPT-2 relies on matrix multiplications and other linear algebra operations that can be easily parallelized.

In addition, GPT-2 has been optimised for distributed training using techniques such as model parallelism and data parallelism. Model parallelism involves splitting the model across multiple devices, while data parallelism involves splitting the input data across multiple devices. By combining these techniques, GPT-2 can be trained on extremely large datasets using clusters of GPUs or other accelerators.

Overall, the parallelizability of GPT-2 makes it an extremely powerful and efficient language model for a wide range of natural language processing tasks, including language generation, language translation, and text summarization. However, effective parallel training requires careful attention to issues such as data distribution, synchronisation, and communication, and is an ongoing area of research in the field of deep learning.

LITERATURE SURVEY

P. Mishra, C. Diwan, S. Srinivasa and G. Srinivasaraghavan (2021) [1] The paper proposes a method for generating titles using a pre-trained transformer model that is fine-tuned on a corpus of news articles. The authors use BERT and GPT-2 models and experiment with different input representations and decoding strategies. However, the method faces challenges such as variability in the length of input text, generating diverse titles for the same input, evaluating performance against human-generated titles, and potential bias in pre-trained models. The paper presents a detailed and well-motivated approach for title generation, with experimental results demonstrating the effectiveness of the proposed method. The analysis sheds light on important aspects of the title generation task.

Abdul Moeed, Yang An, Gerhard Hagerer, and Georg Groh. 2020 [2]. The paper titled "Evaluation Metrics for Headline Generation Using Deep Pre-Trained Embeddings" faces some obstacles such as a constrained dataset, a lack of human assessment, insufficient clarification of embedded technology, and incomplete observations. To address these issues, the authors suggest a new set of evaluation metrics that can provide a more precise evaluation of the system's performance

Kanungo, Y.S., Negi, S., & Rajan, A. (2021)[3]. The creation of advertisement headlines through the utilisation of a self-critical masked language model has encountered obstacles including data bias, ambiguity, computational complexity, and limited generalisation. However, despite these challenges, the technique was able to surpass the baseline in regards to relevance, uniqueness, and maintaining brand consistency

P. Li, J. Yu, J. Chen and B. Guo (2021) [4] The paper presents a proposed model for generating news headlines called HG-News, which is based on a Generative Pre-Training Model. However, the model faces challenges such as ambiguity, data quality, and evaluation. The study observed that the generated headlines from the proposed model are considered to be more informative and readable compared to those generated by the baselines.

Jin, Rong & Hauptmann, Alexander. [5] The paper introduces a new proposed model for title generation, which is a Probabilistic Model. However, the model faces challenges such as limited data availability, increased model complexity, potential unethical title generation, and difficulties in evaluation. The study observed that the proposed model is capable of generating readable titles that are understandable for humans.

Sethi, Nandini & Agrawal, Prateek & Madaan, Vishu & Singh, Sanjay & Kumar, A.. (2016) [6] The paper presents a proposed model for generating titles in the English language using natural language processing techniques such as lexical analysis, part of speech tagging, discourse analysis, and frequency analysis of lexemes. However, the model faces challenges such as the absence of adequate training data, difficulties in contextual understanding, and cross-lingual challenges. The study observed a high level of accuracy in generating relevant titles. Furthermore, the proposed model provides a time and effort-saving process for generating titles.

N. Fatima, A. S. Imran, Z. Kastrati, S. M. Daudpota and A. Soomro (2022) [7] The study conducted a comprehensive review of literature on text generation utilizing deep neural network models, which primarily focused on a proposed deep neural network model. However, the model faces various challenges such as the unethical generation of text. Additionally, the evaluation process and observations present difficulties due to the diverse aspects explored in text generation. Furthermore, various quality metrics are employed to assess the quality of generated texts.

Y. Hayashi and H. Yanagimoto, "Title Generation with Recurrent Neural Network(2016) [8] The paper introduces a proposed methodology for generating titles using a Recurrent Neural Network model. However, the model faces challenges such as vocabulary limitation, limited availability of

trained data, and bias in training data. The study observed that the proposed model is successful in generating appropriate titles in most cases, but in some instances, it may generate random titles.

Putra, Jan Wira Gotama & Khodra, Masayu Leylia. (2017) [9] The paper approach of using pre-trained transformer models, specifically BERT, for automated title generation of learning resources and pathways has the potential to improve the quality of educational content organisation while reducing the time required for its development. While operating these models can be challenging and time-consuming, the benefits of utilising them include the ability to generate relevant and engaging titles for educational materials, which can enhance the overall learning experience for students. By automating the process of title generation, educators and content creators can focus on other aspects of educational content development, thereby increasing efficiency and productivity.

Putra, J. W. G., & Khodra, M. L. (2017)[10].The process of automatic title generation in scientific articles using a summarization approach and natural language processing techniques is a promising tool for authorship assistance. This method specifically addresses the challenges of dealing with domain-specific language and addressing ambiguity to generate titles that are comparable in quality to actual titles. This approach is particularly useful for non-native English speaking authors who may struggle with generating appropriate titles for their articles. By automating the title generation process, authors can save time and effort, while improving the overall quality of their articles. This method has the potential to facilitate the publication of high-quality scientific articles and enhance the scientific communication process.

| SN | PaperTitle/Journal Details | Method/Algorithm | Challenges | Observation |
|-----|---|---|---|---|
| [1] | Automatic Title Generation for Text with Pre-trained Transformer Language Mode. | fine-tuning | variability of the length of input text, evaluating the performance of the generated titles against human-generated titles. | The methods achieves state-of-the-art results on two benchmark datasets (PubMed and CNN/Daily Mail) |
| [2] | Evaluation Metrics for Headline Generation Using Deep Pre-Trained Embeddings | Semantic Similarity Score (SSS), GPT-2 and ULMFiT | Limited dataset, NO human evaluation, Limited explanation of embedded technology | New set of evaluation metrics proposed which provides more accurate evaluation of system performance. |
| [3] | Ad Headline Generation using Self-Critical Masked Language Model | self-critical masked language model (SC-MLM). | Data bias, ambiguity, Computational complexity, Limited generalisation | The method outperformed the baselines in terms of relevance, uniqueness, and brand consistency. |
| [4] | HG-News: News Headline Generation Based on a Generative Pre-Training Model | Generative pretrained model | Ambiguity, Data quality, evaluation | The generated headlines are perceived as more informative and readable than those generated by the baselines. |

| | | | | |
|------|---|---|---|---|
| [5] | A New Probabilistic Model for Title Generation | Probabilistic model | Data availability, model's complexity, unethical title generation, evaluation problem | The model is able to create readable titles for humans. |
| [6] | AUTOMATED TITLE GENERATION IN ENGLISH LANGUAGE USING NLP | Natural language processing techniques lexical analysis, part of speech tagging, discourse analysis, frequencies of lexemes | Lack of training data, contextual understanding, cross-lingual challenges. | High level of accuracy in generating relevant titles. Time and effort saving process. |
| [7] | A Systematic Literature Review on Text Generation Using Deep Neural Network Model | Deep neural network model | Unethical text generation, Problem in evaluation. | Different aspects of text generation are explored. Diverse quality metrics are applied to evaluate texts. |
| [8] | Title Generation with Recurrent Neural Network | Recurrent Neural Network model | Vocabulary limitation, limited availability of trained data, bias in training data | Proposed model generates appropriate titles in most of the cases but in some cases generates random titles. |
| [9] | Automatic Title Generation for Learning Resources and Pathways with Pre-trained Transformer Models | Pre-trained transformer models(BERT) | Time consuming, difficult to operate | The approach presented has the potential to save time and improve the quality of educational content organisation. |
| [10] | Automatic Title Generation in Scientific Articles for Authorship Assistance: A Summarization Approach | Summarization method, Natural language processing techniques | Dealing with domain related domain specific language. Addressing ambiguity. | the system can generate titles that are comparable in quality to the actual title. Useful for authors(non-native english) who struggle with generating titles |

Proposed Methodology:

This paper title generator system involves several key steps, including loading and preprocessing the data, training the model, and generating titles.

To begin, the dataset containing abstracts and titles of papers must be loaded using a library like pandas. Once the data is loaded, it needs to be preprocessed in order to prepare it for training. This involves using a tokenizer to tokenize the abstracts and titles, splitting them into words and subwords. The sequences are then tokenized and padded to ensure that they have the same length, which is necessary for training the model.

After the data is preprocessed, the next step is to train the model. This typically involves using a pre-trained language model like GPT-2, which is a neural network model that has been trained on a large corpus of text data. The pre-trained model can then be fine-tuned on the dataset of abstracts and titles to generate accurate titles for newspapers.

To train the model, a DataLoader method is used to handle batching and shuffling of the data. The model is set to training mode, and a collate function is defined to handle padding of the data. Once the model is trained, it can be saved for further testing.

The final step in building a paper title generator is to generate titles for newspapers. This involves using the pre-trained model to predict a title for a given abstract. A `generate_titles()` method is defined to generate titles from the abstracts, which takes a new abstract, tokenizer, and model as input and returns the generated title as output. The generated title can then be displayed to the user on the frontend.

Overall, building a paper title generator requires a combination of data preprocessing, model training, and prediction using a pre-trained language model. With the right tools and techniques, it is possible to automatically generate accurate titles for papers based on their abstracts.

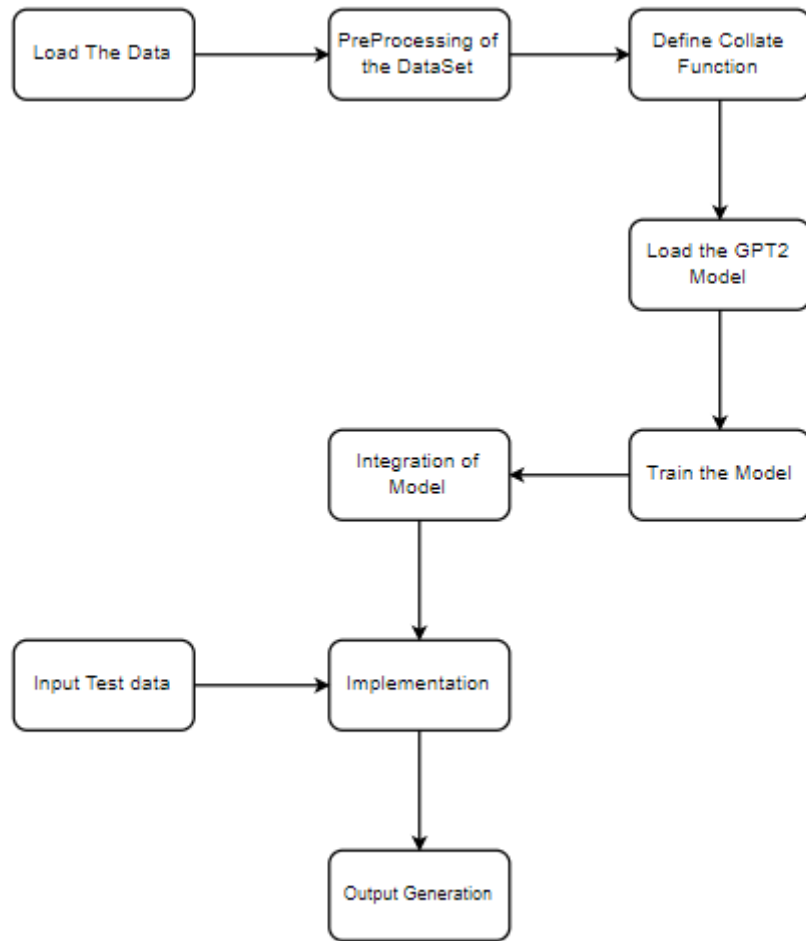


Fig 3.1: Block Diagram of Proposed methodology

4.1. Architecture Diagram:

The proposed system is based on GPT2 model and training the arxiv data set . The trained model is later tuned and called through sub modules of the project for the implementation. GPT-2 (Generative Pre-trained Transformer 2) is a deep learning language model that consists of a multi-layered transformer-based neural network. The architecture of GPT-2 can be divided into several layers or components, each of which plays a crucial role in the model's ability to generate coherent and relevant text. In this answer, we will describe in detail each of the components or layers of GPT-2.

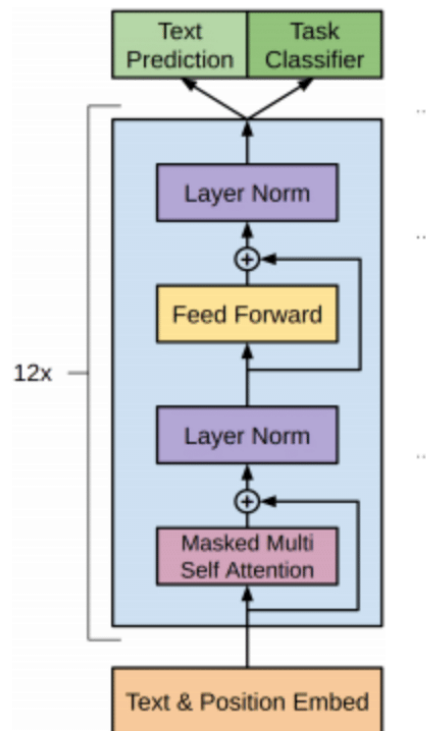


Fig 4.1: Architecture of GPT 2 model [\[11\]](#)

The architecture of GPT-2 can be broken down into three main components:

Decoder: The decoder is responsible for generating new text based on the learned representations from the encoder. It takes the final hidden state from the encoder as input and generates a probability distribution over the vocabulary for each token in the output sequence. During training, the decoder is conditioned on a target sequence of tokens that is shifted by one position, so that the model is trained to predict the next token in the sequence given the previous tokens.

Output Layer: The output layer is a fully connected layer that maps the final hidden state from the decoder to a probability distribution over the entire vocabulary. The token with the highest probability is selected as the predicted next token in the sequence.

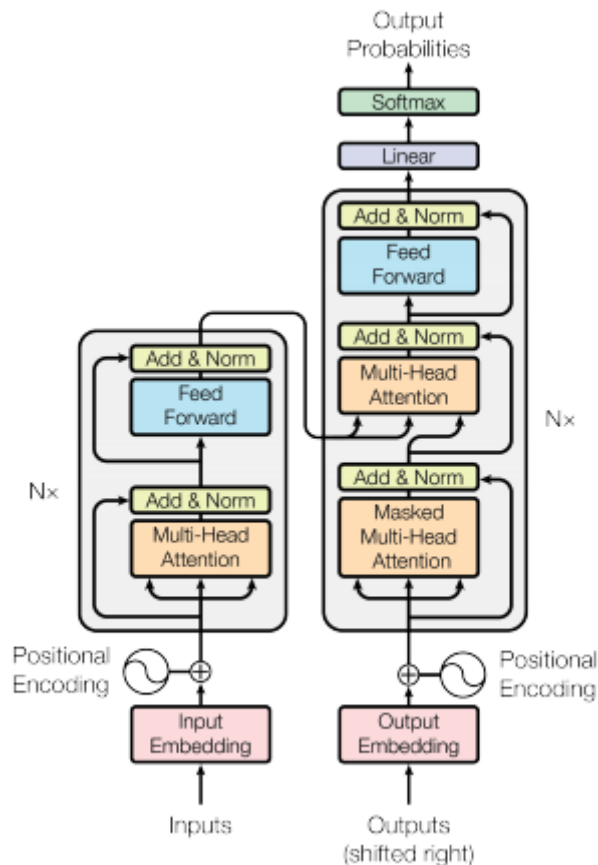


Fig 4.2: Layers in GPT2 [\[11\]](#)

Layers:

Tokenization and Embedding Layer:

The first component of GPT-2 is the tokenization and embedding layer. This layer is responsible for converting the input text into a sequence of discrete tokens and then mapping these tokens to high-dimensional vectors (embeddings) that the model can work with. The tokenization process breaks the text down into individual words or subwords, which are then mapped to a unique integer value that represents the token's position in the model's vocabulary. The embedding layer then maps each integer value to a dense vector in a high-dimensional embedding space. These embeddings capture semantic and syntactic information about the tokens and their relationships to other tokens in the sequence.

Multi-Head Self-Attention Layer:

The multi-head self-attention layer is a critical component of the transformer-based architecture used by GPT-2. This layer allows the model to attend to different parts of the input sequence simultaneously and learn contextual information for each token. The self-attention mechanism works by computing a weighted sum of the embeddings of all tokens in the sequence, where the weights are learned based on the relevance of each token to the current token. This allows the model to focus on relevant parts of the input sequence and ignore irrelevant or redundant information.

Feed-Forward Layer:

After computing self-attention, the model passes the attended embeddings through a feed-forward neural network (FFN) layer. The FFN consists of two linear transformations with a non-linear activation function (such as ReLU) in between. This allows the model to learn complex non-linear relationships between the input tokens and generate more expressive representations.

Layer Normalization:

Layer normalization is a technique used to stabilize the training of deep neural networks by normalizing the input to each layer. This helps to ensure that the distribution of inputs to each layer is consistent, making it easier for the model to learn meaningful representations.

Residual Connections:

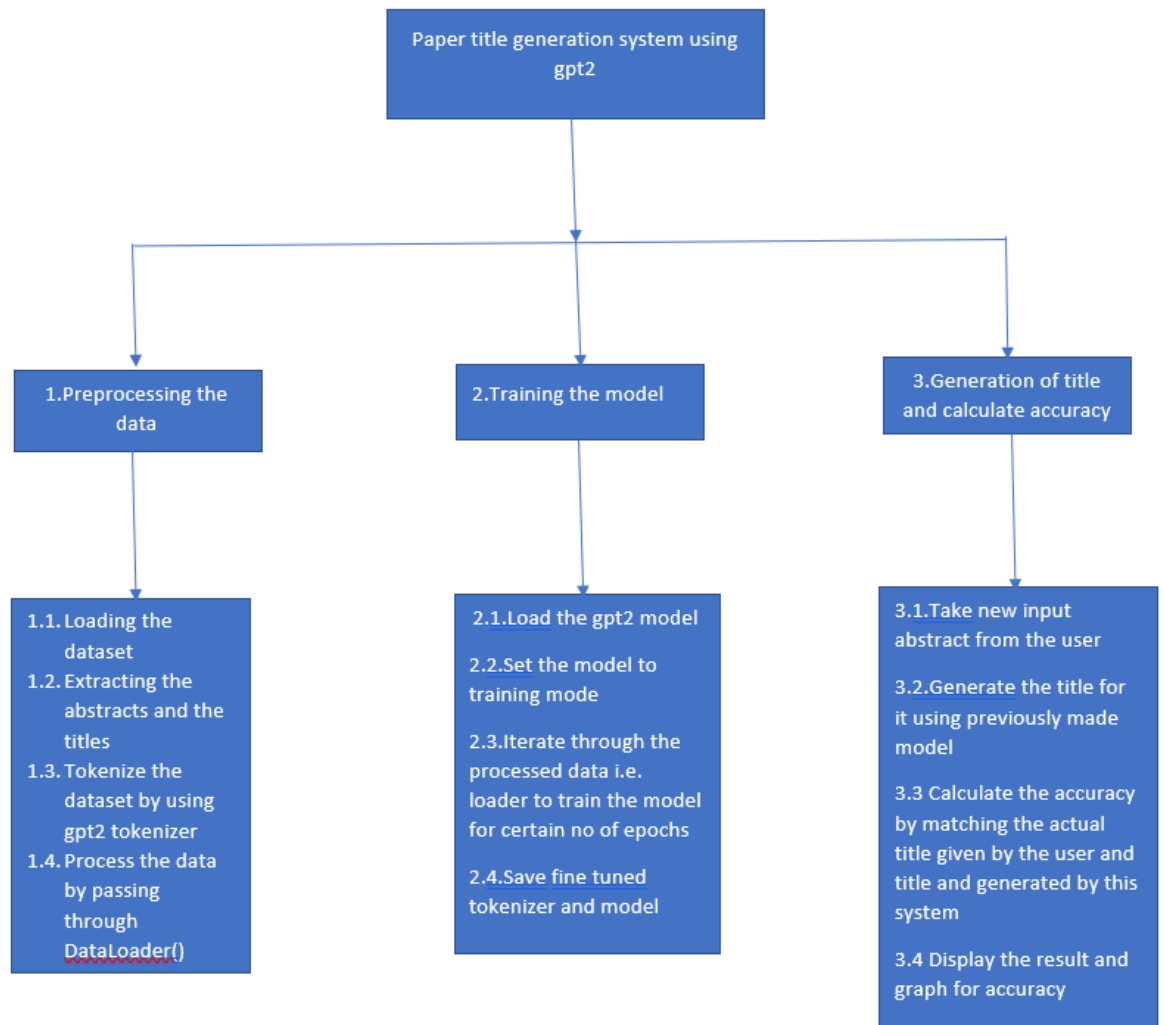
Residual connections are another technique used to improve the stability of deep neural networks. They allow the model to bypass one or more layers and feed the input directly to the next layer. This helps to alleviate the vanishing gradient problem, which can occur when training deep networks.

Positional Encoding:

Positional encoding is a technique used to incorporate the position of each token in the input sequence into the model's representations. This allows the model to distinguish between tokens based on their position, which is critical for generating coherent and contextually appropriate text.

Output Layer:

The final layer of GPT-2 is the output layer, which is responsible for predicting the next token in the sequence. The output layer is a fully connected layer that maps the final hidden state of the model to a probability distribution over the entire vocabulary. The token with the highest probability is selected as the predicted next token in the sequence.



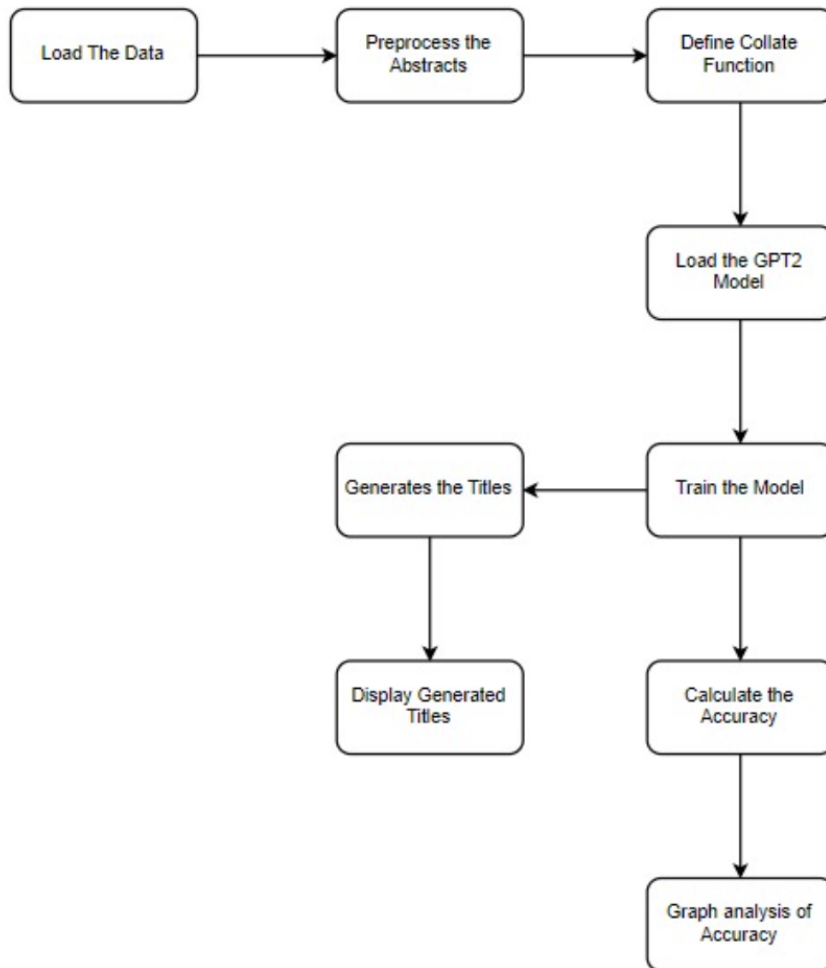
The Proposed Methodology can be viewed in 3 sub modules:

1. Preprocessing the data - It involves loading the dataset, extracting the abstracts and titles. Then tokenizing the dataset by using gpt2 tokenizer. Then processing the data by passing through the `DataLoader()` method.

2. Training the model- It involves loading the gpt2 model, setting the model to training mode, then iterating through the processed data to train the model for a certain number of epochs and finally saving the model.

3. Generation of title and accuracy calculation- It involves taking new input abstract from the user, generation of title using previously trained model. Accuracy is calculated by matching the actual title given by the user and title generated by this system. Finally the result is displayed along with a graph for accuracy.

4.2.Flow Diagram:



1. Loading the data: The dataset containing abstracts and titles of papers is read using pandas and then abstracts and titles are loaded from the CSV file.
2. Data preprocessing: Tokenizer is loaded and abstract and titles are passed to AbstractTitleDataset() class to preprocess the data. Sequences are splitted into words and subwords and tokenized and padded to be passed to train the model. Then the collate function is defined which further tokenizes and adds padding.
3. The DataLoader: DataLoader method is called to handle batching and shuffling of the data. It returns loader which is used for training the model.
4. Load the pre-trained model gpt2: GPT2 model is loaded and set to training mode.

5. Define the collate function: Define a collate function to handle padding of the data. This function stacks the input_ids, attention_mask, and labels for each batch.
6. Train the model : loader from preprocessed data is used to train the gpt2 model and the model is saved for further testing.
7. Generate the titles: generate_titles() method is defined to generate titles from the abstracts. The function takes new abstract, tokenizer and the model as argument and returns the title for the given abstract as output.
8. Display the generated titles: When the user enters the abstract and clicks generate title from the frontend the abstract is passed to generate_titles() method and title is shown to the user at frontend.
9. Display accuracy score: When the user has the actual title for the paper and he/she wants to check the accuracy of title generated by the model , the user can give the actual title of the project and see the accuracy score.

4.3. Pseudo code and explanation:

Pseudocode:

import necessary libraries

define batch_size, num_workers, shuffle, and pin_memory

define AbstractTitleDataset class

- initialize the abstracts, titles, tokenizer, and max_length
- define __len__ function that returns the length of the abstracts
- define __getitem__ function that returns inputs and labels

define collate_fn function

- initialize the tokenizer
- pad the input_ids and attention_mask
- pad the labels
- return a dictionary of input_ids, attention_mask, and labels

if __name__ == '__main__':

- load the dataset from CSV file
- initialize tokenizer and dataset
 - define dataloader using DataLoader class with dataset, batch_size, collate_fn, num_workers, shuffle, pin_memory, and drop_last
- load the model and tokenizer
- set up the optimizer and training loop
- train the model using dataloader and optimizer
- save the model and tokenizer

- define `generate_title` function that takes `abstract`, `tokenizer`, and `model` as input and returns `title`
- generate title using the function and print it

Explanation:

The code defines a PyTorch dataset class called `AbstractTitleDataset` that takes in two arrays of strings - `abstracts` and `titles`, a `tokenizer`, and a maximum sequence length. The class returns a dictionary that contains the input IDs, attention mask, and labels for each sample.

A `collate` function called `collate_fn` is also defined to handle padding of the batch. The function uses PyTorch's `pad_sequence` method to pad the input IDs and attention masks to the same length. It also pads the labels with a value of -100.

The `DataLoader` class is then used to load the dataset in batches. The `batch_size` parameter specifies the number of samples per batch, while the `num_workers` parameter specifies the number of subprocesses to use for data loading. The `shuffle` parameter shuffles the dataset at each epoch, and the `pin_memory` parameter enables fast data transfer to the GPU if available. The code then loads the pre-trained GPT2 language model and tokenizer from the `transformers` library. An Adam optimizer is defined with a learning rate of $1e-5$, and the training loop runs for five epochs. During each epoch, the data loader is iterated over, and the batch is sent to the GPU if available. The optimizer's gradients are zeroed, and the loss is calculated using the model's loss output. The loss is then back propagated, and the weights are updated using the optimizer. The loss is printed every 50 batches.

The trained model is then saved, and a function called `generate_title` is defined to generate a title from an abstract. The function takes in an `abstract`, `tokenizer`, and `model`, and returns a `title`. The function encodes the abstract using the tokenizer, generates the title using the model's `generate` method, and decodes the output using the tokenizer's `decode` method.

Finally, an example usage of the `generate_title` function is shown by generating a title from a new abstract.

5. Experiment and Results

5.1 Data set (Sample with Explanation)

ARXIV DATASET:

ArXiv is a repository of scientific papers in the fields of physics, mathematics, computer science, quantitative biology, quantitative finance, and statistics. The ArXiv dataset contains metadata and abstracts of all papers submitted to the repository since its inception in 1991.

The ArXiv dataset is available for download in several formats, including raw text, XML, and JSON. The dataset includes information such as the title of the paper, the authors, the abstract, the date of submission, the subject categories, and the full text of the paper in some cases.

The ArXiv dataset is a valuable resource for researchers and data scientists working in the fields of physics, mathematics, computer science, and other related fields. The dataset can be used for a wide range of tasks, such as text classification, topic modelling, citation analysis, and recommendation systems.

One of the challenges of working with the ArXiv dataset is its size. As of 2021, the dataset contains over 1.7 million papers and grows at a rate of about 10,000 papers per month. Processing and analyzing such a large dataset can be computationally intensive and require significant resources.

Another challenge is the quality of the data. The dataset contains papers submitted by authors from around the world, and the quality and accuracy of the data may vary. Some papers may contain errors, inconsistencies, or inaccuracies that can affect the results of data analysis.

Despite these challenges, the ArXiv dataset remains a valuable resource for researchers and data scientists working in the fields of physics, mathematics, and computer science. Its large size and rich metadata make it an excellent resource for studying scientific trends, developing new machine learning models, and discovering new scientific insights.

Metadata of the Dataset:



This dataset is a mirror of the original ArXiv data. Because the full dataset is rather large (1.1TB and growing), this dataset provides only a metadata file in the json format. This file contains an entry for each paper, containing:

- id: ArXiv ID (can be used to access the paper, see below)
- submitter: Who submitted the paper
- authors: Authors of the paper
- title: Title of the paper
- comments: Additional info, such as number of pages and figures
- journal-ref: Information about the journal the paper was published in
- doi: [<https://www.doi.org/>](Digital Object Identifier)
- abstract: The abstract of the paper
- categories: Categories / tags in the ArXiv system
- versions: A version history

Sample Dataset:

The dataset is fetched from the Arxiv data set. tHere is the sample of the data in a json format contained in the dataset.

Refer: <https://www.kaggle.com/datasets/Cornell-University/arxiv>

```
▼ "root" : { 14 items 
  "id" : string "0704.0001"
  "submitter" : string "Pavel Nadolsky"
  "authors" : string "C. Bal'azs, E. L. Berger, P. M. Nadolsky, C.-P. Yuan" 
  "title" : string "Calculation of prompt diphoton production cross sections at Tevatron and LHC energies"
  "comments" : string "37 pages, 15 figures; published version"
  "journal-ref" : string "Phys.Rev.D76:013009,2007"
  "doi" : string "10.1103/PhysRevD.76.013009"
  "report-no" : string "ANL-HEP-PR-07-12"
  "categories" : string "hep-ph"
  "license" : NULL
  "abstract" :
    string " A fully differential calculation in perturbative quantum chromodynamics is presented for the production
    of massive photon pairs at hadron colliders. All next-to-leading order perturbative contributions from quark-
    antiquark, gluon-(anti)quark, and gluon-gluon subprocesses are included, as well as all-orders resummation of
    initial-state gluon radiation valid at next-to-next-to-leading logarithmic accuracy. The region of phase space
    is specified in which the calculation is most reliable. Good agreement is demonstrated with data from the
    Fermilab Tevatron, and predictions are made for more detailed tests with CDF and DO data. Predictions are
    shown for distributions of diphoton pairs produced at the energy of the Large Hadron Collider (LHC).
    Distributions of the diphoton pairs from the decay of a Higgs boson are contrasted with those produced from
    QCD processes at the LHC, showing that enhanced sensitivity to the signal can be obtained with judicious
    selection of events. "
```

5.1.1 Explain the methodology with the dataset. – (Proof of results derived from the dataset)

Data processing:

With the data set we will fetch the required data i.e title abstract using different required filters like duration of year, author, category and many more. Here we use inbuilt libraries numpy, pandas to fetch the required data set and store them in a dataframe and finally store them as a csv file which is going to be used by training model.

- Fetch required data from the arxiv dataset

```
[6]: import json

data_file = '../input/arxiv/arxiv-metadata-oai-snapshot.json'

def get_metadata():
    with open(data_file, 'r') as f:
        for line in f:
            yield line

[7]: metadata = get_metadata()
for paper in metadata:
    paper_dict = json.loads(paper)
    print('Title: {}\n\nAbstract: {}\n\nRef: {}'.format(paper_dict.get('title'), paper_dict.get('abstract'), paper_dict.get('journal-ref')))
    # print(paper)
    break
```

Title: Calculation of prompt diphoton production cross sections at Tevatron and LHC energies

Abstract: A fully differential calculation in perturbative quantum chromodynamics is presented for the production of massive photon pairs at hadron colliders. All next-to-leading order perturbative contributions from quark-antiquark, gluon-(anti)quark, and gluon-gluon subprocesses are included, as well as all-orders resummation of initial-state gluon radiation valid at next-to-next-to-leading logarithmic accuracy. The region of phase space is specified in which the calculation is most reliable. Good agreement is demonstrated with data from the Fermilab Tevatron, and predictions are made for more detailed tests with CDF and DO data. Predictions are shown for distributions of diphoton pairs produced at the energy of the Large Hadron Collider (LHC). Distributions of the diphoton pairs from the decay of a Higgs

- Filter the dataset according to the year, category, author as required.

```
titles = []
ids = []
abstracts = []
authors_parsed = []
authors = []
years = []
metadata = get_metadata()
for paper in metadata:
    paper_dict = json.loads(paper)
    ref = paper_dict.get('journal-ref')
    try:
        year = int(ref[-4:])
        if 2018 < year < 2022:
            years.append(year)
            ids.append(paper_dict.get('id'))
            authors_parsed.append(paper_dict.get('authors_parsed'))
            authors.append(paper_dict.get('authors'))
            titles.append(paper_dict.get('title'))
            abstracts.append(paper_dict.get('abstract'))
    except:
        pass

len(titles), len(abstracts), len(years)
```

[8]: (16834, 16834, 16834)

- Make a data frame of the fetched data set

```
papers = pd.DataFrame({
    'id' : ids,
    'title': titles,
    # 'authors': authors,
    # 'authors_parsed': authors_parsed,
    'abstract': abstracts,
    'year': years
})
papers.head()
```

```
[9]:
```

| | id | title | abstract | year |
|---|-----------|---|---|------|
| 0 | 0804.3104 | Teichmüller Structures and Dual Geometric Gi... | The Gibbs measure theory for smooth potentia... | 2020 |
| 1 | 0807.3139 | Weight Reduction for Mod I Bianchi Modular Forms | Let K be an imaginary quadratic field with c... | 2019 |
| 2 | 0810.5491 | Nonequilibrium phase transition in a spreading... | We consider a nonequilibrium process on a ti... | 2020 |
| 3 | 0812.2682 | Quantum integrable systems in three-dimensiona... | In this paper we construct integrable three-... | 2019 |
| 4 | 0901.0172 | Numerical Performance of Compact Fourth Order ... | In this study the numerical performance of t... | 2019 |

- Convert the data into downloadable csv file

```
[11]: papers.to_csv("nlp.csv")
```

+ Code

+ Markdown

Output (19.4MB / 19.5GB)

▼ /kaggle/working



📄 nlp.csv

Refer: <https://www.kaggle.com/code/aniswi2002/transformers-generating-titles-from-abstracts/edit>

Training:

Well tokenized and padded data is then used to train the model. Every instance of the dataset is visited and used to train the model in every epoch.

```
for epoch in range(num_epochs):
    print("Epoch:", epoch + 1)
```

```
# Set the model to training mode
model.train()
```

```
# Iterate over the data loader
for batch_idx, batch in enumerate(loader):
    # Send the batch to the GPU if available
    input_ids = batch['input_ids'].cuda()
```

```

attention_mask = batch['attention_mask'].cuda()
labels = batch['labels'].cuda()

# Zero the gradients
optimizer.zero_grad()

# Calculate the loss
outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
loss = outputs.loss

# Backpropagate the loss and update the weights
loss.backward()
optimizer.step()

# Print the loss every 50 batches
if batch_idx % 50 == 0:
    print("Batch:", batch_idx, "Loss:", loss.item())

# Save the model
model.save_pretrained("my_finetuned_gpt2_model")
tokenizer.save_pretrained("my_finetuned_gpt2_tokenizer")

```

Here loader is the preprocessed data which is traversed and used to train the model for each epoch.

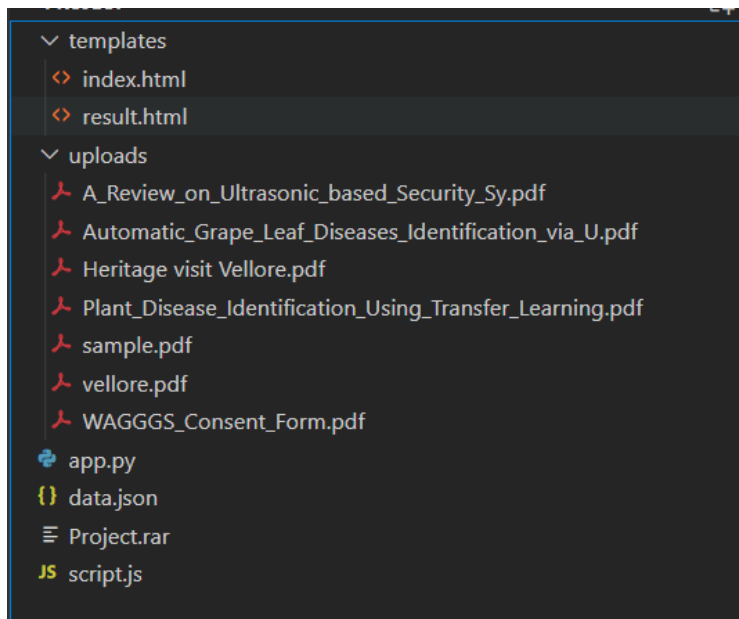
Implementation:

We integrate our trained model with web applications for better understanding and interaction. The pre-trained model is fetched through the backend where flask is used and to get and post the value from the user frontend using html, css and js is used. The accuracy is also calculated throughout the paper by providing its actual title so that we can match how accurate the trained model is. The accuracy is stored in a json file which is used to make a graph of the stored accuracy with id.

Backend : app.py , data.json

Frontend: templates {index.html, result.html, script.js}

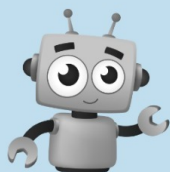
Assets: upload



5.2 Sample Output screen



Generated Title

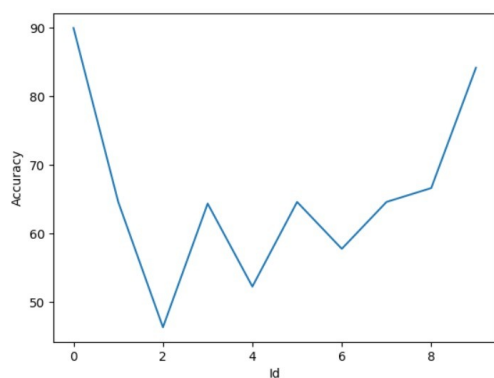
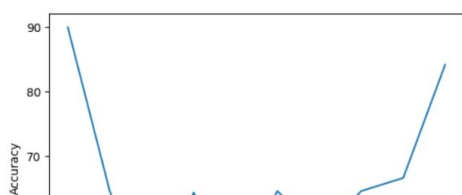


Title:

A Review on Ultrasonic based Security System

Accuracy:

84.21052631578947



Conclusion and Future work

Conclusion:

In this project, we have demonstrated Flask, OpenAI, PyPDF2, and Matplotlib to build a web application that generates titles for scientific papers based on their content. We leveraged the power of OpenAI's GPT (Generative Pre-trained Transformer) to generate high-quality titles that are similar to the actual titles of scientific papers. We have also shown how to evaluate the accuracy of the generated titles and plot a chart of the accuracy values over time.

Our results demonstrate the effectiveness of using GPT-based models for natural language generation tasks, even with limited training data. Future work could explore ways to further fine-tune GPT models on domain-specific scientific language to improve the accuracy of generated titles.

Future Work:

There are several directions for future work that can be pursued to improve this application. First, we can explore different natural language processing techniques to further improve the accuracy of the generated titles. Second, we can integrate this application with other tools for scientific research, such as reference managers or citation generators, to provide a more comprehensive workflow for researchers. Third, we can expand the scope of this application to include other types of documents, such as reports, proposals, or grant applications. Finally, we can develop a user feedback mechanism to gather more data on the accuracy of the generated titles and use it to fine-tune the system.

References:

1. P. Mishra, C. Diwan, S. Srinivasa and G. Srinivasaraghavan, "Automatic Title Generation for Text with Pre-trained Transformer Language Model," *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, Laguna Hills, CA, USA, 2021, pp. 17-24, doi: 10.1109/ICSC50631.2021.00009.
2. Abdul Moeed, Yang An, Gerhard Hagerer, and Georg Groh. 2020. [Evaluation Metrics for Headline Generation Using Deep Pre-Trained Embeddings](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1796–1802, Marseille, France. European Language Resources Association.
3. Kanungo, Y.S., Negi, S., & Rajan, A. (2021). Ad Headline Generation using Self-Critical Masked Language Model. *North American Chapter of the Association for Computational Linguistics*.
4. P. Li, J. Yu, J. Chen and B. Guo, "HG-News: News Headline Generation Based on a Generative Pre-Training Model," in *IEEE Access*, vol. 9, pp. 110039-110046, 2021, doi: 10.1109/ACCESS.2021.3102741.
5. Jin, Rong & Hauptmann, Alexander. (2003). A New Probabilistic Model for Title Generation. 10.3115/1072228.1072365.
6. Sethi, Nandini & Agrawal, Prateek & Madaan, Vishu & Singh, Sanjay & Kumar, A.. (2016). Automated title generation in the English language using NLP. 9. 5159-5168.

7. N. Fatima, A. S. Imran, Z. Kastrati, S. M. Daudpota and A. Soomro, "A Systematic Literature Review on Text Generation Using Deep Neural Network Models," in *IEEE Access*, vol. 10, pp. 53490-53503, 2022, doi: 10.1109/ACCESS.2022.3174108.
8. Y. Hayashi and H. Yanagimoto, "Title Generation with Recurrent Neural Network," *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, Kumamoto, Japan, 2016, pp. 250-255, doi: 10.1109/IIAI-AAI.2016.109
9. Putra, Jan Wira Gotama & Khodra, Masayu Leylia. (2017). Automatic Title Generation in Scientific Articles for Authorship Assistance: A Summarization Approach. *Journal of ICT Research and Applications*. 11. 253. 10.5614/itbj.ict.res.appl.2017.11.3.3.
10. Putra, J. W. G., & Khodra, M. L. (2017). Automatic Title Generation in Scientific Articles for Authorship Assistance: A Summarization Approach. *Journal of ICT Research and Applications*, 11(3), 253-267. <https://doi.org/10.5614/itbj.ict.res.appl.2017.11.3.3>
11. [https://datascience.stackexchange.com/questions/85486/what-is-the-difference-between-gp
t-blocks-and-transformer-decoder-blocks](https://datascience.stackexchange.com/questions/85486/what-is-the-difference-between-gpt-blocks-and-transformer-decoder-blocks)