October University for Modern Sciences and Arts

Faculty of Computer Science

Graduation Project

Generating Scientific Paper Titles

Supervisor: Dr. Wael Hassan Gomaa

Name: Andrew Melad Makram Girgis

ID: 202011

Table of Contents

List of Figures:

List of tables:

Table of abbreviations:

Abbreviation	Word
ACL	Association for Computational Linguistics
ASPT	ArXiv Scientific Paper Titles
BART	Bidirectional and Auto-Regressive Transformers
BERT	Bidirectional Encoder Representations from Transformers
IDE	Integrated Development Environment
LDC	Linguistic Data Consortium
LSTM	Long Short-Term Memory networks
mBART	multilingual Bidirectional and Auto-Regressive Transformers
NLP	Natural Language Processing
PSPT	PubMed Scientific Paper Titles
RNN	Recurrent Neural Networks
SDG	Sustainable Development Goals
SVMs	Support Vector Machines
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
BLEU	Bilingual Evaluation Understudy

Abstract

Much research has been done for generating scientific papers titles for many years, and the creation of headlines is a crucial duty in text summarization. Although there is an infinite quantity of training data available for this task, it is nevertheless difficult since learning to generate article headlines requires a model with excellent natural language capabilities. A collection of documents' content is sometimes summarized briefly in the headline of an article reporting on a particular occurrence. Because creating a headline by hand is neither inexpensive nor timely, knowing how to generate a relevant headline automatically is essential. Our main target is to generate scientific titles from papers. In my project, we are using dataset which called arXiv dataset. The dataset contains 19174 records, 15339 records used as training data and 3835 records used as testing data.

.

Chapter 1: Introduction

1.1 Introduction

My project belongs to number four in `Sustainable Development Goals (SDG) which is quality education. Natural Language Processing is a part of computer science, human language, and artificial intelligence that focuses on enabling machines to understand [1]. NLP techniques involve a range of tasks, including language translation, sentiment analysis, text classification, information extraction, speech recognition, headline generation, and dialogue systems. These tasks are performed by applying machine learning and deep learning. NLP has many practical applications, including chatbots, language translation, virtual assistants, sentiment analysis for marketing and social media, and automated customer service systems. Our project which is generating scientific paper titles focuses on natural language processing (NLP), which is the study of teaching computers to effortlessly grasp human language much like humans. Generating scientific paper titles is very important step in research process. A creative title can attract readers, explain the main idea of the research, and help it stand out among the thousands of articles published every year. The title should be short, accurate, and informative, giving a brief summary of the paper's content. The study question, the key findings, and the target audience must all be taken into consideration when coming up with an engaging scientific paper title. While using suitable terminology and staying away from language that can be confusing to non-experts, the title should be informative and related to the field. Also, it is important to ensure that the title accurately summarizes the paper's content and represents the findings. Finally, generating scientific paper titles that will be helpful requires finding a dynamic equilibrium between being accurate and representative of the research material while also being informative and interesting. Researchers may successfully communicate their findings and raise the profile and effect of their work by using a well-written title. Researchers have been investigating the use of machine learning techniques to generate scientific paper titles automatically as a solution to this problem. The use of algorithms in machine learning involves discovering patterns in data and applying those patterns to inform predictions or decisions. Machine learning algorithms can be taught on a large corpus of scientific papers in the context of generating titles for scientific papers. The use of machine learning algorithms for generating scientific paper titles is the focus of this discussion. Machine learning techniques are becoming more popular for generating scientific paper titles. Many strategies, including the use of transformer models or recurrent neural networks (RNNs), have been suggested by numerous studies. Machine learning techniques can be used to generate scientific paper titles, which has several advantages including reducing the time and effort needed to write a title and increasing its correctness and relevance. So far, there are some disadvantages too, such as the incoherence of the titles generated and the potential of bias in the training data. The research on how to address these problems and raise the quality of the titles that are generated while preserving their accuracy and relevance is missing [2].

1.2 Problem statement

The challenge faced by researchers in coming up with a short and informative title that accurately reflects the content of their research is the issue to be solved in terms of generating scientific paper titles. Creating a successful title requires the author to reduce their work's content into a short phrase or sentence, which can be difficult, especially for non- native English speakers. Also, the title ought to be catchy, highlighting the importance of the study and stimulate readers' interest. Automated methods that can produce superior scientific paper titles that conform to these standards and effectively convey the core of the research are required to solve this issue. Such tools would be especially helpful for researchers and non-native English speakers who have experience in creating catchy titles.

1.3 Objective

This project should present a fully developing machine learning model that can generate scientific paper titles based on the content of the paper is the goal of this research. The titles that are generated should accurately reflect the main contribution of the research, be informative, and be brief. The research objectives of this project are collect and preprocess a large collection of scientific title and papers from many fields, develop and implement a deep learning architecture that can identify the semantic connections between the paper's content and its related title, to improve the performance of the model, experiment with various hyper - parameters and architectural designs, analyze the model's output in great detail, paying particular attention to the accuracy, coherence, and uniqueness of the titles that are generated, and analyze the limitations and advantages of the suggested model and specify areas that could want improvement in future research.

1.4 Motivation

The suggested research on machine learning-based title generation for scientific papers is very significant in a variety of fields, including academics, business, and research institutions. For the effective transmission and distribution of research findings, short and accurate titles are important. The suggested research can contribute to enhancing the discoverability and accessibility of scientific publications as well as the quality of their titles, all of which can promote scientific information. The proposed project provides a chance for researchers to make a contribution to machine learning and natural language processing. It can have significant effects on many research areas and offer valuable information about the relationship between title and content to develop a successful model for generating scientific paper titles.

1.5 Thesis layout

We have included an abstract, an introduction, an overview, and the project's objectives in the first chapter. The backdrop of the relevant work that was carried out in the same field of study will then be covered in the second chapter's literature review. In-depth information about the data used, the tools and libraries we used, our working environment, and the suggested system architecture will all be covered in the third chapter.

\

Chapter 2: Background and Literature Review

2.1 Background

Generating scientific paper titles is a difficult task that requires knowledge with the language used in the field as well as the ability to communicate professionally the core of the research. To address this task, a variety of approaches and algorithms have been developed, including rule-based systems, statistical models, and neural networks. In rule-based systems, the semantic and syntactic patterns of scientific titles are encoded using established rules. For example, a rule-based system might suggest a title "Assessing the Impacts of Climate Change on Biodiversity in Coastal Areas" for a paper on climate change. Machine learning techniques are used by statistical models to identify patterns from a corpus of scientific papers. These models are capable of capturing complex relationships between a title's words and the paper's content. For example, a statistical model may suggest a title for a paper on cancer research as "Exploring the Role of Genetic Variants in Cancer Risk: A Large-Scale Genome-Wide Association Study". Deep learning algorithms are used by neural network-based models to automatically identify the patterns and structures in scientific titles. The underlying semantic content of the research work is captured by these models, which can generate titles of a high-quality. For instance, a machine learning article might have the title "Deep Reinforcement Learning for Intelligent Agent Navigation in Dynamic Environments" suggested by a neural network-based model [3].

Table 1: Examples of generated paper titles using different approaches [3].

Approach	**Example Paper Title**
Rule-based	Uncovering the Role of Microbes in Soil Health
Statistical	Investigating the Effect of Climate Change on Crop Yield
Neural Network	Learning to Detect Anomalies in Network Traffic

2.1.1 Machine learning approach

An interesting challenge that may be tackled using a variety of methods is coming up with titles for scientific papers using machine learning. Careful data preparation, feature engineering, and model selection are required when building a machine learning model for generating scientific paper titles. It is possible to come up with accurate and meaningful titles that can be helpful to researchers and scientists with the correct methodology and methods. To produce precise and relevant titles, it's essential that you choose the right machine learning model. Decision trees, random forests, and support vector machines are a few of the well-liked models for natural language processing applications **(SVMs)** [4].

2.1.2 Deep learning approach

It is an interesting and challenging procedure that may be tackled in a variety of ways to generate the titles of scientific papers using deep learning. Data collection, data pre-processing, model selection, model training, and model evaluation are a few of the stages involved in developing this task utilizing deep learning. It's critical to choose the best deep learning model in order to provide precise and pertinent titles. Several models are used for natural language processing tasks, including recurrent neural networks (RNNs), LSTMs, and transformer models like BERT or GPT-2 or T5 [5].

2.2 Previous Work

2.2.1 Research 1: BERT: pre-training of deep bidirectional transformers for language understanding (2018).

- In this research, the BERT model of language representation is introduced. It is an NLP machine learning framework that is available for free. To help computers, understand the meaning of ambiguous words in text, it uses the surrounding text to provide context. Language models could only read input from the right to the left or the left to the right, not both at once. Bidirectionality refers to the ability to read simultaneously in both directions. BertSumAbs model is used. The encoder is constructed using pretrained Bidirectional Encoder Representations from Transformers. As pretrained BERT for BertSumAbs, using news data and RuBert, which was trained on the Russian section of Wikipedia. This data was also used to create a 120K-word vocabulary of Russian sub-tokens. BertSumAbs has 320M parameters [6].

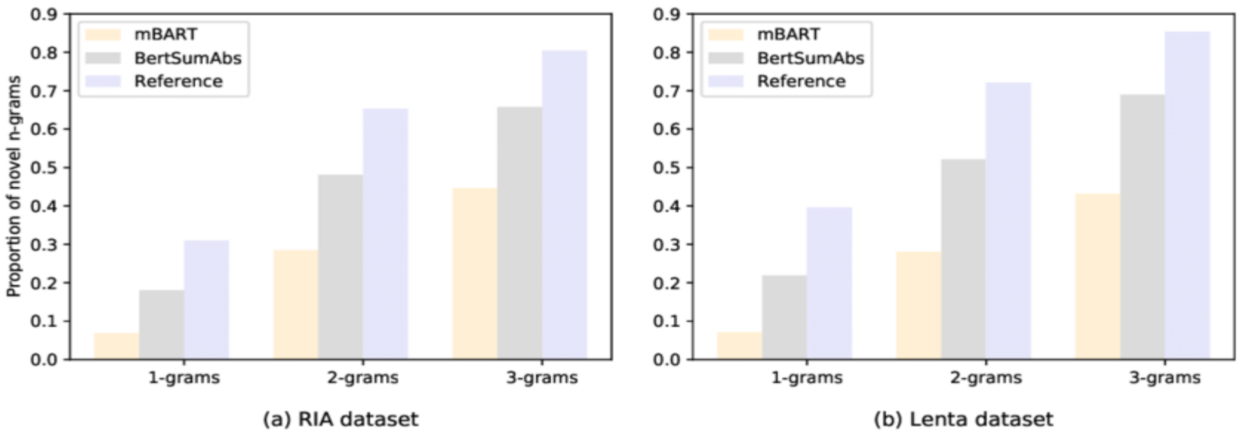


Figure1: Result of n-gram using two models with two different datasets. *[6]*

2.2.2 Research 2: Multilingual denoising pre-training for neural machine translation (2020).

- This research has applied transformer-based models follows some steps one of them is supervised training. In this research, **mBART** model is pre-trained language model, which was developed by Facebook AI Research. It is used that is standard transformer-based consisting of an encoders and autoregressive decoders. It is a development of the well-known **BART** model, which was developed using just English-language monolingual data. On the other hand, the mBART model was trained on parallel data in 25 distinct languages, including, among others, Arabic, Chinese, French, German, Japanese, Russian, and Spanish. A subset of 25 languages is used to pretrain mBART once for all languages. With a total of around 680M parameters and 12 encoder and 12 decoder layers, it has a vocabulary of 250K words. Supervised training is one of the procedures that come before using Transformer-based models [7].

Chapter 3: Material and Methods

3.1 Materials

3.1.1 Data

There are numerous datasets that are well-liked for the process of creating scientific article titles, starting with **Scisumm-corpus-master** dataset that contains summaries of academic papers in the field of computational linguistics. The dataset was created to be used in research on the automatic summary of scientific papers. The dataset consists of 1400 scientific papers from the ACL Anthology, a collection of articles about computational linguistics. The dataset for each paper contains a summary of the paper as well as metadata about the authors, the location of the publication, and the year of publication. The expert annotators who have been requested to write the summaries were asked to highlight the paper's major contributions. On the website of the **LDC** (Linguistic Data Consortium), the dataset can be downloaded. Please keep in mind though, that access to a dataset requires a license and is not free [8]. **ASPT** dataset more than 1.7 million scientific paper titles from the ArXiv database are included in this dataset, which covers a wide range of scientific fields. The ArXiv website provides a download for the dataset, ACL-ARC Scientific Paper Corpus this dataset includes full-text articles from the Computational Linguistics journal and the Association for Computational Linguistics **(ACL)** Anthology. The dataset includes the full text of the papers as well as metadata as authors, abstracts, and publishing details, **PSPT** dataset more than 25 million titles of scientific papers from the PubMed database are included in this dataset, which covers a wide range of biomedical and life science subjects. From the PubMed website, you can download the dataset, and The **PLOSONE Corpus** is a dataset that contain more than 185,000 scientific papers from the PLOSONE journal that cover many different branches of science. The dataset includes the full text of the papers as well as metadata as authors, abstracts, and publishing details. In my project, we are using a dataset which called **arXiv dataset**. The dataset contains 19174 records, 15339 records used as training data and 3835 records used as testing data.

3.1.2 Dataset Example

Table 2: Dataset example before pre-processing.

Text
This paper considers the effect of least squares procedures for nearly unstable linear time series with strongly dependent innovations. Under a general framework and appropriate scaling, it is shown that ordinary least squares procedures converge to functionals of fractional Ornstein--Uhlenbeck processes. We use fractional integrated noise as an example to illustrate the important ideas. In this case, the functionals bear only formal analogy to those in the classical framework with uncorrelated innovations, with Wiener processes being replaced by fractional Brownian motions. It is also shown that limit theorems for the functionals involve nonstandard scaling and nonstandard limiting distributions. Results of this paper shed light on the asymptotic behavior of nearly unstable long-memory processes.

Table 3: Dataset example after pre-processing.

Text	Title
paper consideration effect of least square procedure nearly unstable linear time series with strongly dependent innovations. under general framework and appropriate scaling, it is shown that ordinary least squares procedures converge to functionals of fractional Ornstein-Hollenbeck processes. We use fractional integrated noise as an example to illustrate the important ideas. In this case, the functionals bear only formal analogy to those in the classical framework with uncorrelated innovations, with wiener processes being replaced by fractional Brownian motions. It is also shown that limit theorems for the functionals involve nonstandard scaling and nonstandard limiting distributions. results of this paper shed light on the asymptotic behavior of nearly unstable long-memory processes.	Least squares for nearly unstable linear time series with dependent innovations

3.1.3 Tools

- An integrated development environment (IDE) for the Python programming language is PyCharm Community Edition 2021.3. It is PyCharm's open-source, free version, which was developed by JetBrains. Python application development, testing, and debugging features such code analysis, debugging tools, code completion, and Git integration are all included in PyCharm Community Edition 2021.3.
- Python 3.9 is the python version used in this project. It is a significant update that brings several new features and improvements to the Python programming language, making the development of high-quality Python code faster and easier.
- The OS library offers tools to make interacting with the operating system easier.
- NLTK library is an open-source that offers a set of tools for tasks like tokenization, tagging, sentiment analysis, and text preprocessing.
- The re module offers a number of regular expression-related functions, including search, match, findall, sub, and split. These operations include locating all instances of a pattern in a string, swapping out a pattern with another string, and breaking a string into substrings depending on a pattern.
- Logging in NLP refers to recording important events and information during program execution for analysis, debugging, and monitoring purposes.
- JSON (JavaScript Object Notation) in NLP is a lightweight data interchange format commonly used for storing and transmitting structured textual data, making it easy to parse and access information.
- NumPy (Numerical Python) in NLP is a powerful library for efficient numerical computations, providing high-performance multidimensional array objects and a collection of mathematical functions to manipulate and analyze textual data.
- ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics commonly used in Natural Language Processing (NLP) for evaluating the quality of automatic summarization systems by comparing generated summaries with reference summaries based on recall and precision measures.

- BLEU (Bilingual Evaluation Understudy) is a metric used in Natural Language Processing (NLP) for evaluating the quality of machine-translated text by comparing it to one or more reference translations, based on n-gram precision and brevity penalty.
- Cosine similarity is a metric used in Natural Language Processing (NLP) to measure the similarity between two vectors representing text documents by computing the cosine of the angle between them, indicating the degree of similarity or relatedness.

3.1.3 Environment

The Environment is working on System 64-bit, local CPU, Intel(R) Core (TM) i7-8565U CPU @1.80GHz, 16.0 GB RAM and Graphics card Nivida GTX1060.

3.2 Methods

3.2.1 System architecture Overview

Seven steps make up the structure of our system. Reading papers is the first step. We are currently working on reading the entire dataset. At the second stage, known as pre-processing, we begin by using some NLP techniques, such as changing all phrases into lower case, getting rid of stop words and special characters, and stemming or lemmatizing each word. The pre-processing stage is among the most crucial ones and is crucial to the effectiveness of the model we've suggested. Sentence boundary detection, which is the third stage, seeks to recognize individual sentences inside a text. Natural language processing tasks such as named entity recognition, part-of-speech tagging, dependency parsing, and machine translation all use sentences as their input unit. The embedding stage is the fourth. Ranking is done in step five. The creation of titles using deep learning or machine learning is the sixth stage. The final step is to evaluate the suggested model based on the title after reading the user input.

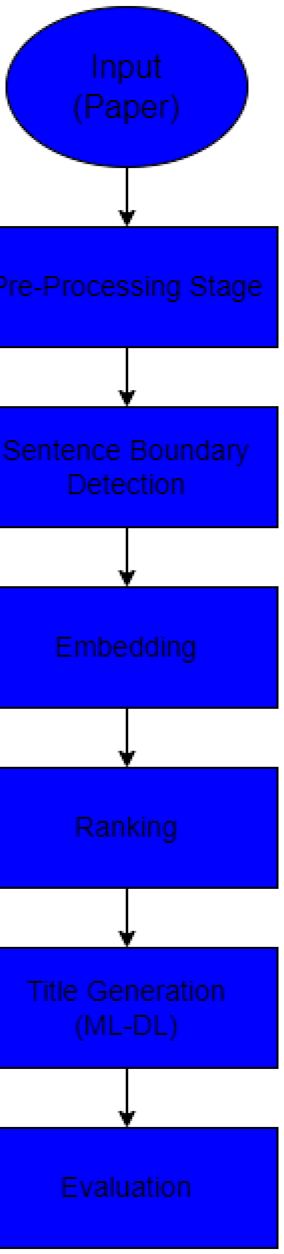


Figure 2: Architecture of Proposed model

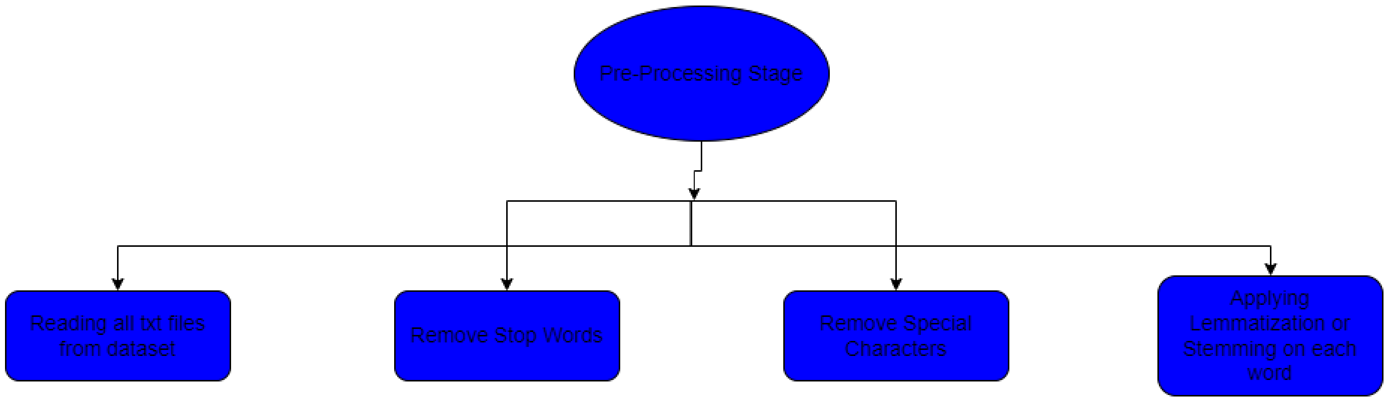

Pre-Processing Stage
Reading all txt files from dataset
Remove Stop Words
Remove Special Characters
Applying Lemmatization or Stemming on each word

Figure 3: Stage of Pre-Processing

According to ‘Figure 6; it consists of a few key NLP approaches throughout the pre-processing stage. The first way involves changing every letter in every text to lower case because a capital "A" does not equal a little "a." The second method is getting rid of stop words. The third method is the elimination of special characters. The final strategy involves stemming or lemmatizing each word in each phrase. Stemming involves returning each word to its original form, whereas lemmatization entails integrating several words' inflected forms into a single unit that can be investigated.

Chapter 4: System Implementation

4.1 System Development:

The system to produce titles from abstracts was constructed using the SVM model. Its development was broken down into numerous stages with the goal of determining how well it would perform this task. To begin with, a small-scale SVM model was used to test its ability to provide insightful and pertinent titles. A subset of abstracts was used to train and test this prototype model, which allowed an evaluation of its performance. During the testing phase, Figure 1 shows the input abstracts and the corresponding generated titles.

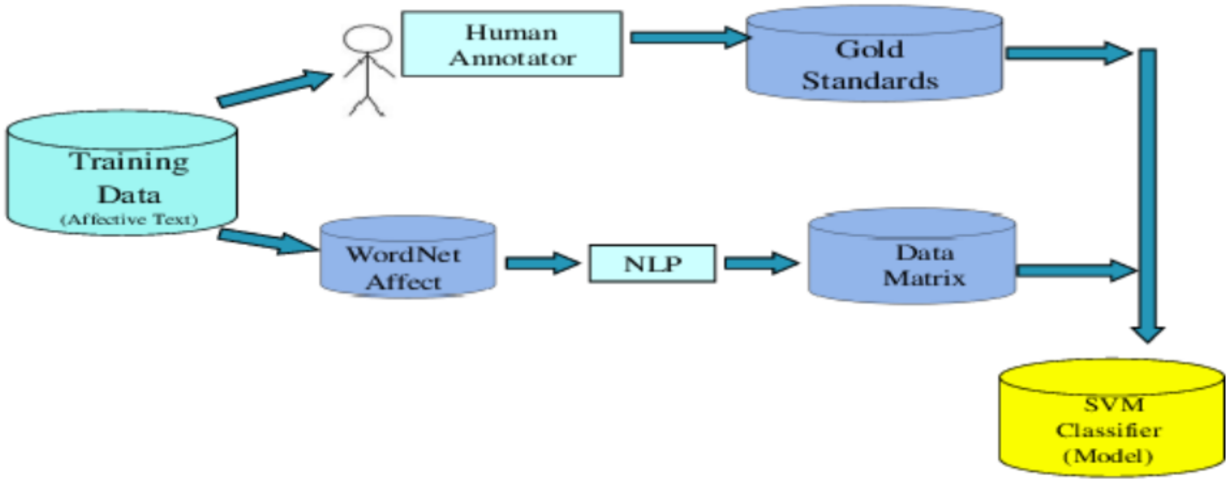


Figure 4: Input abstracts and generated titles for SVM model testing

A helper tool was created to facilitate data preparation after it was determined that SVM was appropriate for title generating. This program made it easier to do tasks including data format conversion, subset extraction, and abstract preprocessing. Its goal was to make sure that the input data was correctly formatted and prepared for the SVM model's training and evaluation. The program greatly simplified the entire development process, allowing for effective data processing. Next, PartY was created using the SVM model to incorporate algorithm. This stage sought to evaluate the algorithm's performance in producing correct and instructive titles from the provided abstracts. The generated titles' accuracy and relevancy were improved by making corrections after comparing the output titles to the anticipated outcomes. Finally, partX and partW were combined with the SVM model to create an improved version of partY. The final planned system for title generation from abstracts was built on this integration. The combination of these elements ensured enhanced performance and raised the system's overall efficacy. Different software platforms were used during the system development to make it easier to apply and assess the SVM model. During various stages of development, platforms like Unity, OpenCV, Matlab, TensorFlow, Keras, Spyder, Jupyter, and Colab were used, each with a specific function. The architecture of the SVM model within the system is shown in Figure 8.

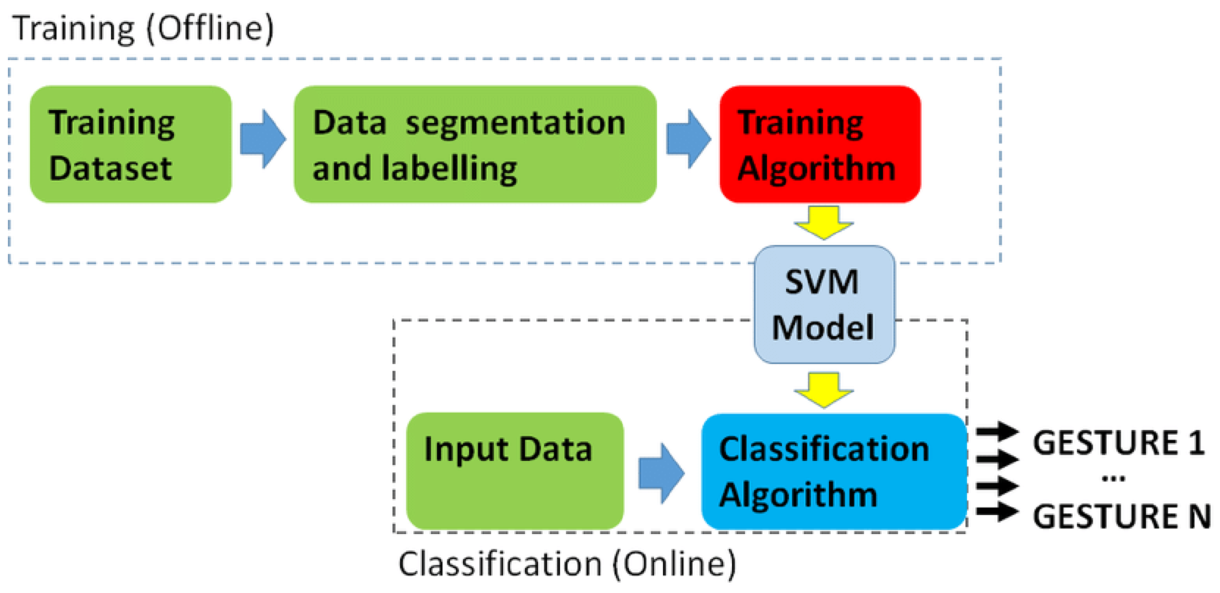
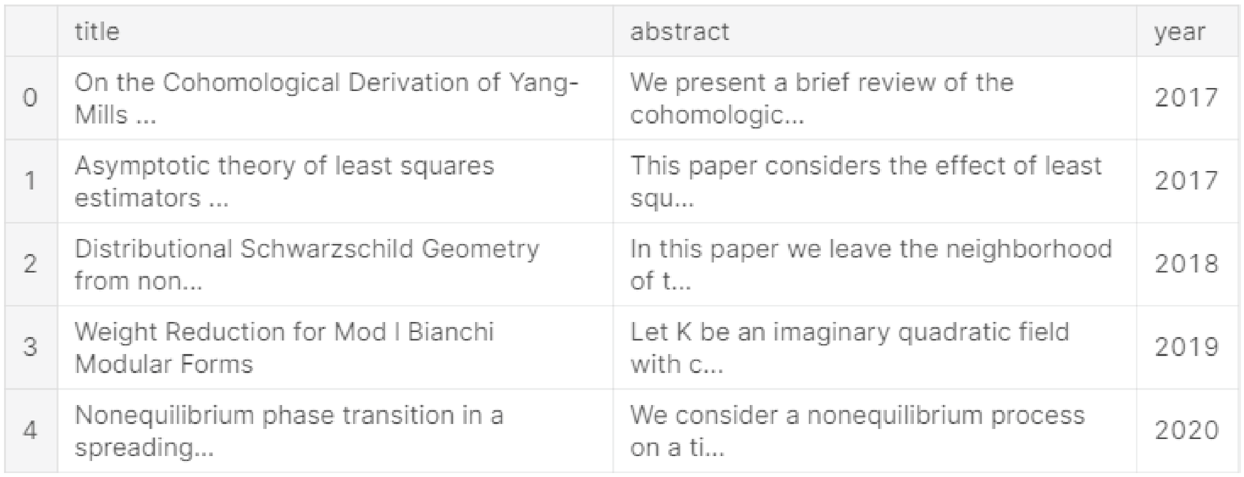


Figure 5: Architecture of the SVM model within the system

In the "Discussion & Conclusion" chapter, the difficulties faced when using each software platform will be thoroughly examined, giving insights into potential restrictions and problems related to its usage.

Simple T5 Model: As part of the system to produce titles from abstracts, a straightforward T5 model was created. It is comparable to the SVM model. The framework of the T5 model's development was similar to that of the SVM model. A small-scale T5 model was initially put into practice to test how well it could produce pertinent titles from abstracts. A portion of the dataset was used to train and validate this prototype model, allowing for an assessment of its performance. The input abstracts and accompanying produced titles used during the T5 model's testing phase are shown in Table 4.

Table 4: Input abstracts and generated titles for T5 model testing.



	title	abstract	year
0	On the Cohomological Derivation of Yang-Mills ...	We present a brief review of the cohomologic...	2017
1	Asymptotic theory of least squares estimators ...	This paper considers the effect of least squ...	2017
2	Distributional Schwarzschild Geometry from non...	In this paper we leave the neighborhood of t...	2018
3	Weight Reduction for Mod l Bianchi Modular Forms	Let K be an imaginary quadratic field with c...	2019
4	Nonequilibrium phase transition in a spreading...	We consider a nonequilibrium process on a ti...	2020

A help tool made especially for the T5 model was developed to guarantee flawless data handling and preprocessing. This program made it easier to format data, extract subsets, and do other essential data operations. The total development process was streamlined by using the assistance tool to adapt the input abstracts to the demands of the T5 model The T5 model was then used to develop portion Y in order to incorporate algorithm Z. This stage attempted to assess how well the algorithm produced precise and educational titles from the provided abstracts. The generated titles' correctness and relevancy were improved by comparing the output titles to the anticipated outcomes. The T5 model was then combined with parts X and W to create an improved version of part Y. The T5 model, partX, and partW were all integrated to create the final proposed system for title creation from abstracts, which improved overall performance. The T5 model's internal design is shown in Figure 4 of the system.

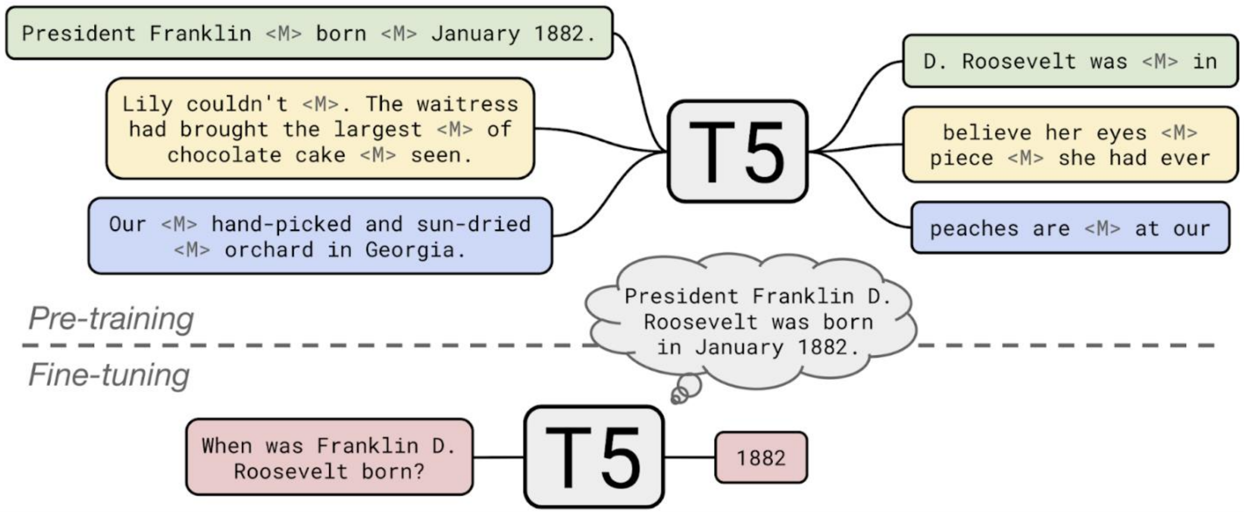

President Franklin <M> born <M> January 1882.
Lily couldn't <M>. The waitress had brought the largest <M> of chocolate cake <M> seen.
Our <M> hand-picked and sun-dried <M> orchard in Georgia.
T5
D. Roosevelt was <M> in
believe her eyes <M> piece <M> she had ever
peaches are <M> at our
President Franklin D. Roosevelt was born in January 1882.
Pre-training
Fine-tuning
When was Franklin D. Roosevelt born?
T5
1882

Figure 6: Architecture of the T5 model within the system

Several software platforms were used in the creation of the T5 model, including Unity, OpenCV, Matlab, TensorFlow, Keras, Spyder, Jupyter, and Colab. These platforms were chosen due to their suitability for various stages of development and their capacity to meet needs. The "Discussion & Conclusion" chapter will elaborate on the difficulties encountered when using each platform, giving readers a thorough grasp of the constraints and problems encountered throughout creation.

BART Model: In a similar vein, the system to produce titles from abstracts was built using the BART model. The BART model was selected mostly because of its capacity to produce excellent textual outputs. The BART model was developed during the course of numerous crucial stages. In the beginning, a small-scale BART model was used to gauge how well it produced accurate and pertinent titles from abstracts. With the help of a portion of the dataset, this basic model was trained, and its performance assessed. The input abstracts and correspondingly generated titles used during the BART model. A companion tool was created to streamline the required activities for data preparation and conversion. This tool made it possible to efficiently convert data formats, extract subsets, and perform preprocessing, guaranteeing that the input abstracts were formatted correctly for the BART model's training and evaluation. The BART model was then used to create component Y and incorporate algorithm Z. This stage intended to evaluate how well the algorithm produced excellent titles from the supplied abstracts.

The generated titles were compared to the anticipated outcomes, and changes were made to improve the precision and coherence of the wording.

4.2 System Structure:

The flow between the components/modules of the final system we constructed will be explained in the following subsections, and the architecture of this automated system will be shown in the second subsection.

4.2.1 System Architecture:

The framework of our system consists of seven steps. The first step is to read articles. The complete dataset is now being read as we speak. Pre-processing is the second stage, where we start by removing stop words and special characters, converting all phrases into lower case, and stemming or lemmatizing each word. The efficiency of the model we've proposed depends on the pre-processing stage, which is one of the most important ones. The third stage, sentence boundary identification, aims to identify distinct sentences within a document. Sentences are the input unit for several tasks in natural language processing, including named entity recognition, part-of-speech tagging, dependency parsing, and machine translation. The fourth stage is embedding. In step five, the ranking is completed. The sixth stage involves producing titles using deep learning or machine learning. After reviewing the user input, the suggested model is evaluated based on the title.

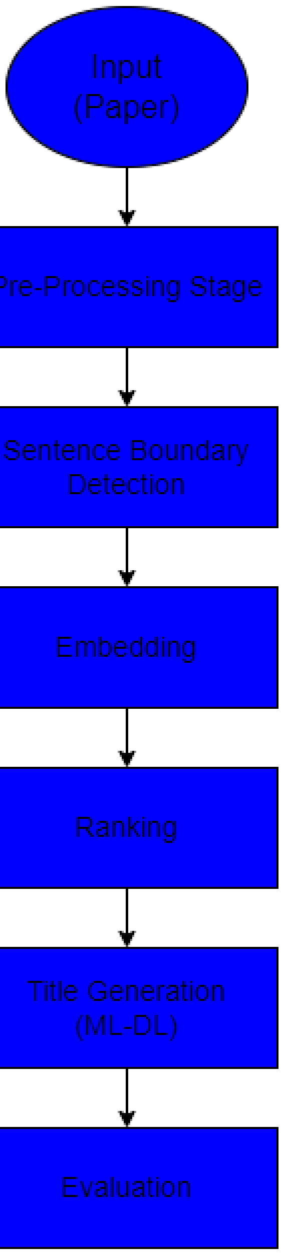


Figure 2: Architecture of Proposed model

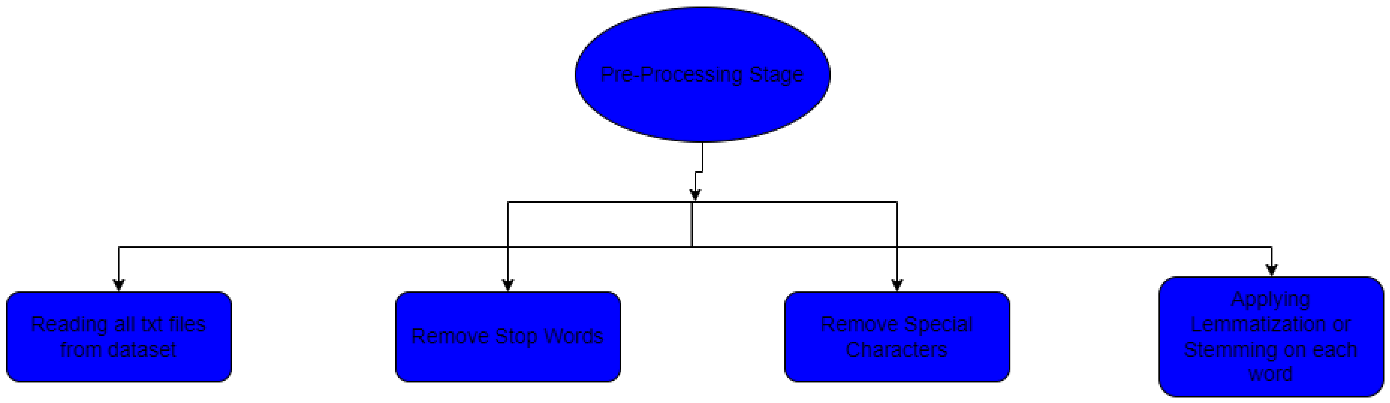

Pre-Processing Stage
Reading all txt files from dataset
Remove Stop Words
Remove Special Characters
Applying Lemmatization or Stemming on each word

Figure 3: Stage of Pre-Processing

Figure 6 shows that it uses a limited number of essential NLP techniques during the pre-processing phase. The first method entails altering every letter in every text to lower case since a capital "A" does not equal a small "a." The second strategy is to eliminate stop words. The next strategy involves getting rid of special characters. The third tactic is lemmatizing or stemming every word in each phrase. Lemmatization includes combining numerous words' inflected forms into a single unit that may be studied, whereas stemming means restoring each word to its original form.

4.2.2 Tensor Board:

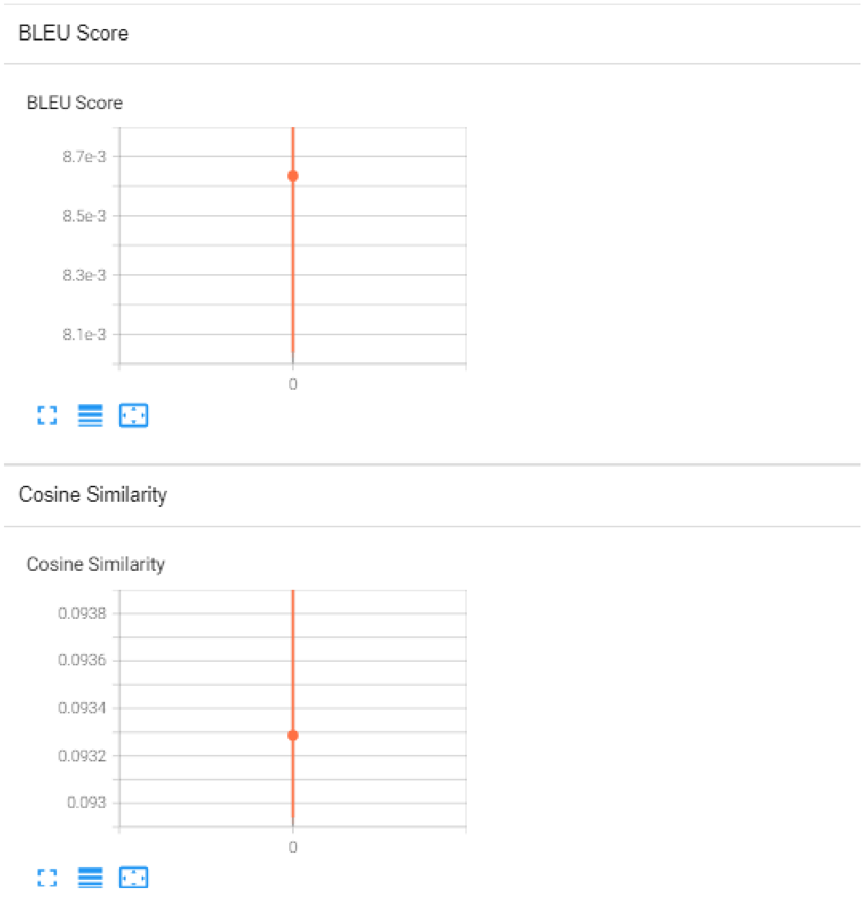


Figure 7: Scalars of SVM model.

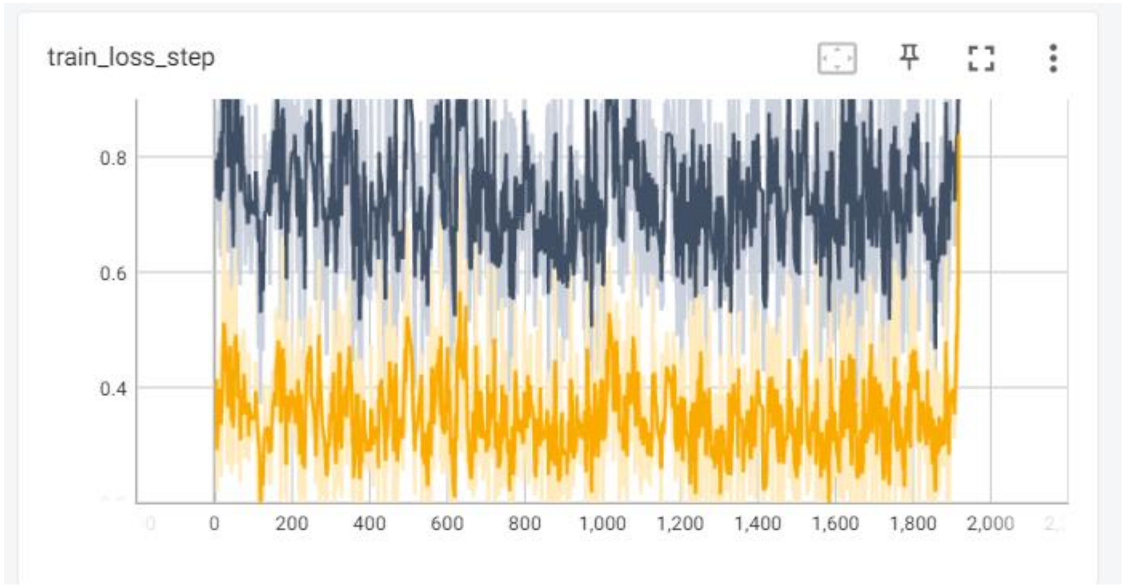


Figure 8: Scalars of Simple T5 model.

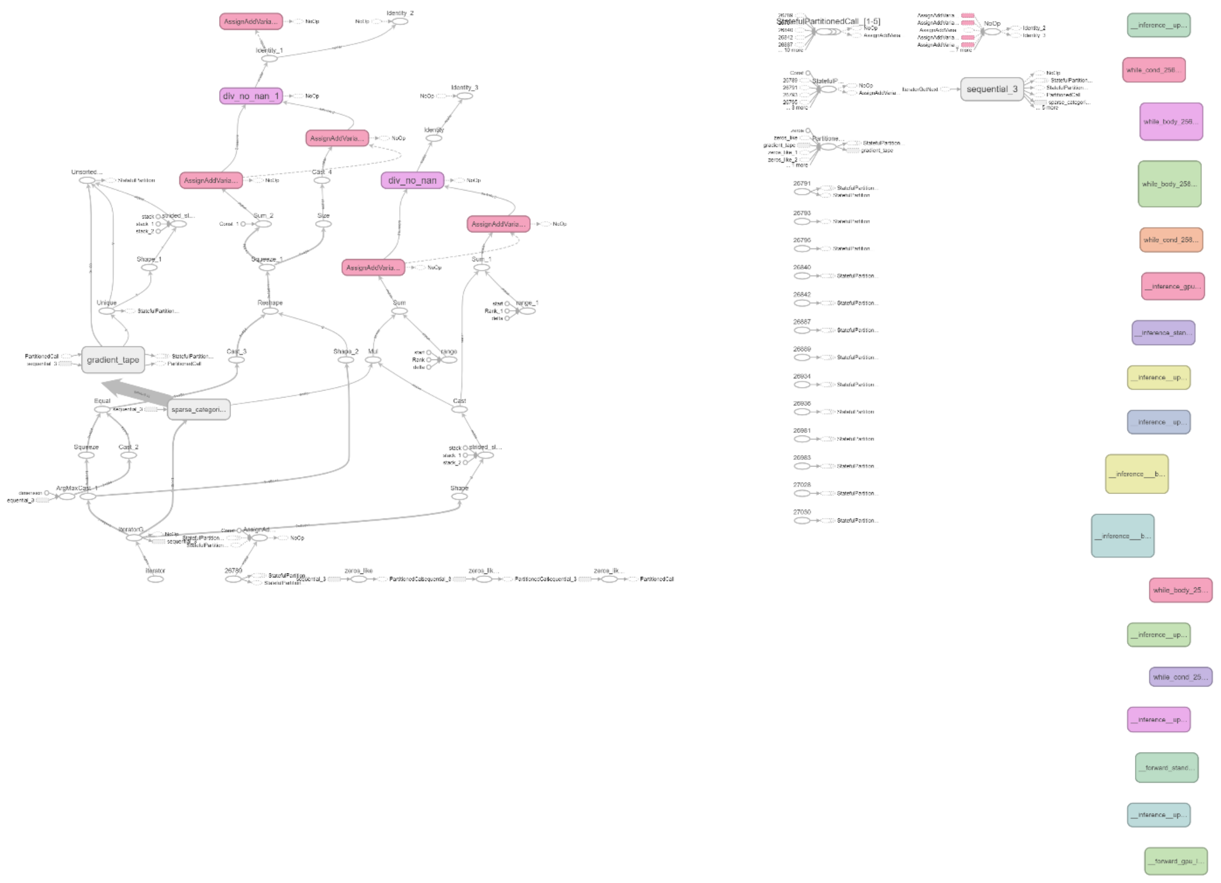
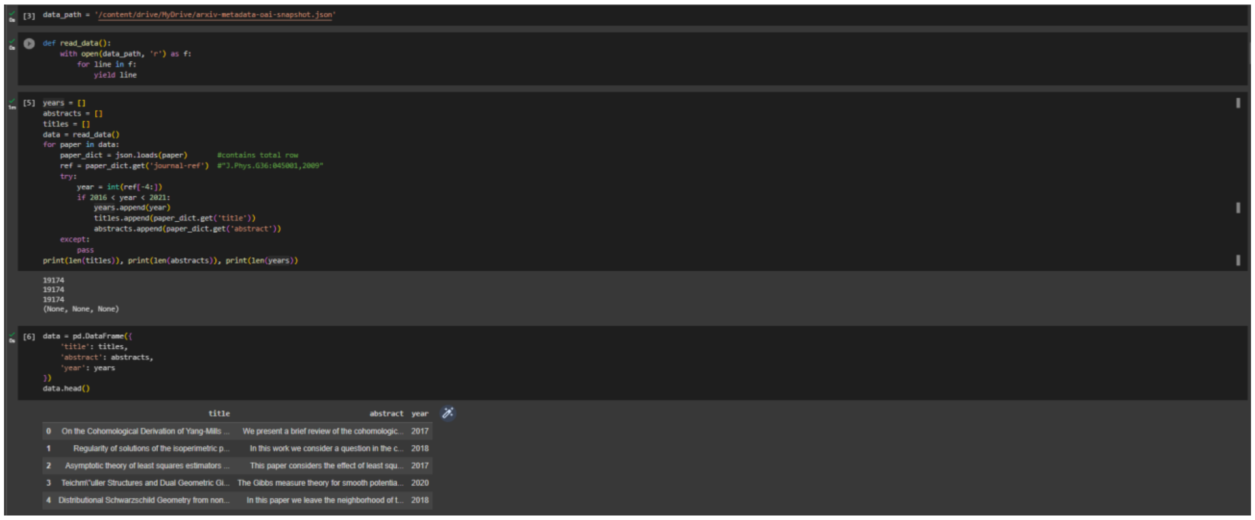


Figure 9: Tensorboard General Arch.

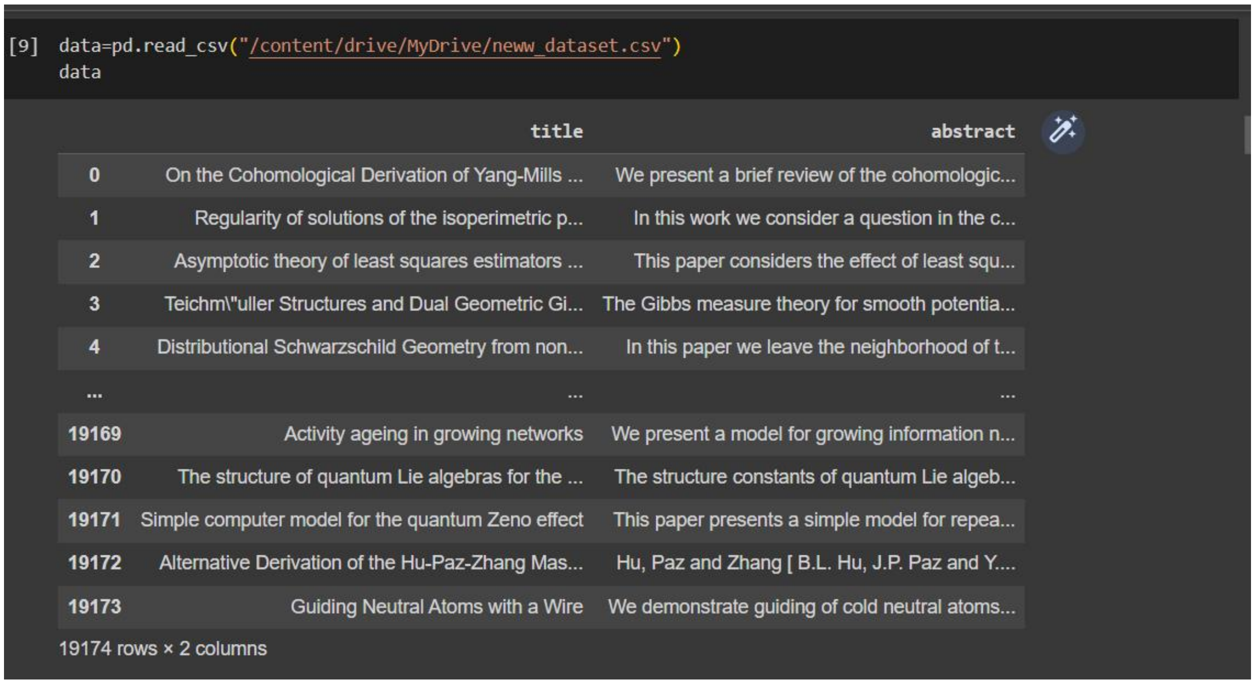
4.3 System Running:



```
[3] data_path = '/content/drive/MyDrive/arxiv-metadata-oai-snapshot.json'

def read_data():
    with open(data_path, 'r') as f:
        for line in f:
            yield line

[5] years = []
abstracts = []
titles = []
data = read_data()
for paper in data:
    paper_dict = json.loads(paper)       #contains total row
    ref = paper_dict.get('journal-ref')  #"J.Phys.G36:045001,2009"
    try:
        year = int(ref[-4:])
        if 2016 < year < 2021:
            years.append(year)
            titles.append(paper_dict.get('title'))
            abstracts.append(paper_dict.get('abstract'))
    except:
        pass
print(len(titles)), print(len(abstracts)), print(len(years))

19174
19174
19174
(None, None, None)

[6] data = pd.DataFrame({
    'title': titles,
    'abstract': abstracts,
    'year': years
})
data.head()
```

	title	abstract	year
0	On the Cohomological Derivation of Yang-Mills ...	We present a brief review of the cohomologic...	2017
1	Regularity of solutions of the isoperimetric p...	In this work we consider a question in the c...	2018
2	Asymptotic theory of least squares estimators ...	This paper considers the effect of least squ...	2017
3	Teichm\"uller Structures and Dual Geometric Gi...	The Gibbs measure theory for smooth potentia...	2020
4	Distributional Schwarzschild Geometry from non...	In this paper we leave the neighborhood of t...	2018

Figure 10: Read the dataset.

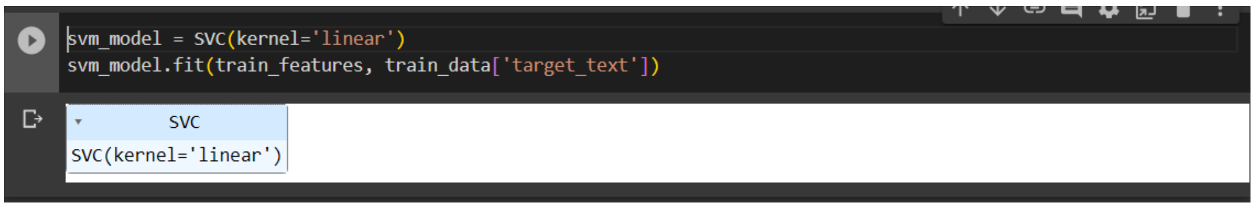


```
[9] data=pd.read_csv("/content/drive/MyDrive/neww_dataset.csv")
    data
```

	title	abstract
0	On the Cohomological Derivation of Yang-Mills ...	We present a brief review of the cohomologic...
1	Regularity of solutions of the isoperimetric p...	In this work we consider a question in the c...
2	Asymptotic theory of least squares estimators ...	This paper considers the effect of least squ...
3	Teichm\"uller Structures and Dual Geometric Gi...	The Gibbs measure theory for smooth potentia...
4	Distributional Schwarzschild Geometry from non...	In this paper we leave the neighborhood of t...
...	...	...
19169	Activity ageing in growing networks	We present a model for growing information n...
19170	The structure of quantum Lie algebras for the ...	The structure constants of quantum Lie algeb...
19171	Simple computer model for the quantum Zeno effect	This paper presents a simple model for repea...
19172	Alternative Derivation of the Hu-Paz-Zhang Mas...	Hu, Paz and Zhang [B.L. Hu, J.P. Paz and Y....
19173	Guiding Neutral Atoms with a Wire	We demonstrate guiding of cold neutral atoms...

19174 rows × 2 columns

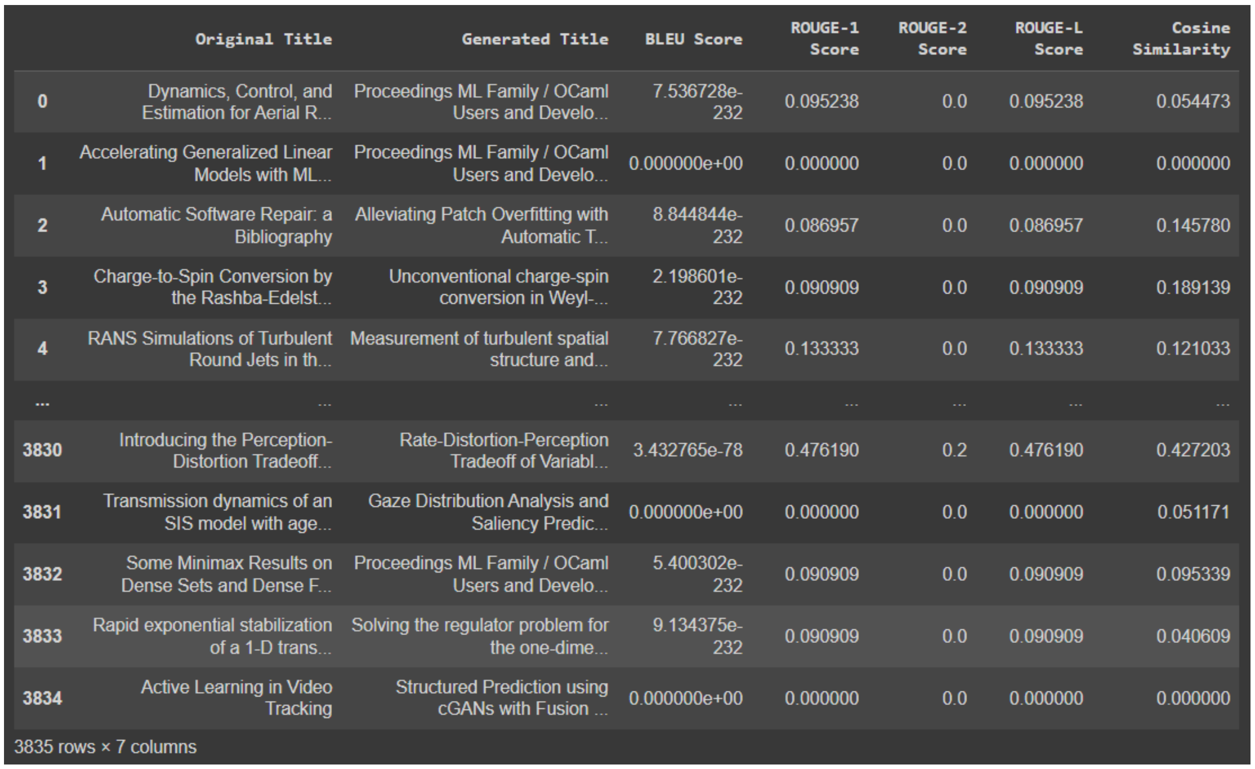
Figure 11: Read the dataset after removing year column.



```
svm_model = SVC(kernel='linear')
svm_model.fit(train_features, train_data['target_text'])
```

```
SVC
SVC(kernel='linear')
```

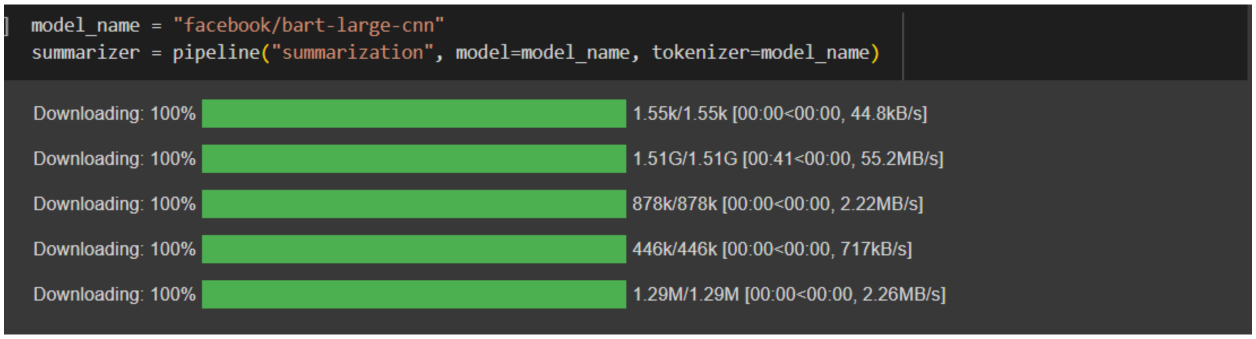
Figure 12: Train the Support Vector Machine (SVM) model.



	Original Title	Generated Title	BLEU Score	ROUGE-1 Score	ROUGE-2 Score	ROUGE-L Score	Cosine Similarity
0	Dynamics, Control, and Estimation for Aerial R...	Proceedings ML Family / OCaml Users and Develo...	7.536728e-232	0.095238	0.0	0.095238	0.054473
1	Accelerating Generalized Linear Models with ML...	Proceedings ML Family / OCaml Users and Develo...	0.000000e+00	0.000000	0.0	0.000000	0.000000
2	Automatic Software Repair: a Bibliography	Alleviating Patch Overfitting with Automatic T...	8.844844e-232	0.086957	0.0	0.086957	0.145780
3	Charge-to-Spin Conversion by the Rashba-Edelst...	Unconventional charge-spin conversion in Weyl-...	2.198601e-232	0.090909	0.0	0.090909	0.189139
4	RANS Simulations of Turbulent Round Jets in th...	Measurement of turbulent spatial structure and...	7.766827e-232	0.133333	0.0	0.133333	0.121033
...	...	...	...	...	...	...	...
3830	Introducing the Perception-Distortion Tradeoff...	Rate-Distortion-Perception Tradeoff of Variabl...	3.432765e-78	0.476190	0.2	0.476190	0.427203
3831	Transmission dynamics of an SIS model with age...	Gaze Distribution Analysis and Saliency Predic...	0.000000e+00	0.000000	0.0	0.000000	0.051171
3832	Some Minimax Results on Dense Sets and Dense F...	Proceedings ML Family / OCaml Users and Develo...	5.400302e-232	0.090909	0.0	0.090909	0.095339
3833	Rapid exponential stabilization of a 1-D trans...	Solving the regulator problem for the one-dime...	9.134375e-232	0.090909	0.0	0.090909	0.040609
3834	Active Learning in Video Tracking	Structured Prediction using cGANs with Fusion ...	0.000000e+00	0.000000	0.0	0.000000	0.000000

3835 rows × 7 columns

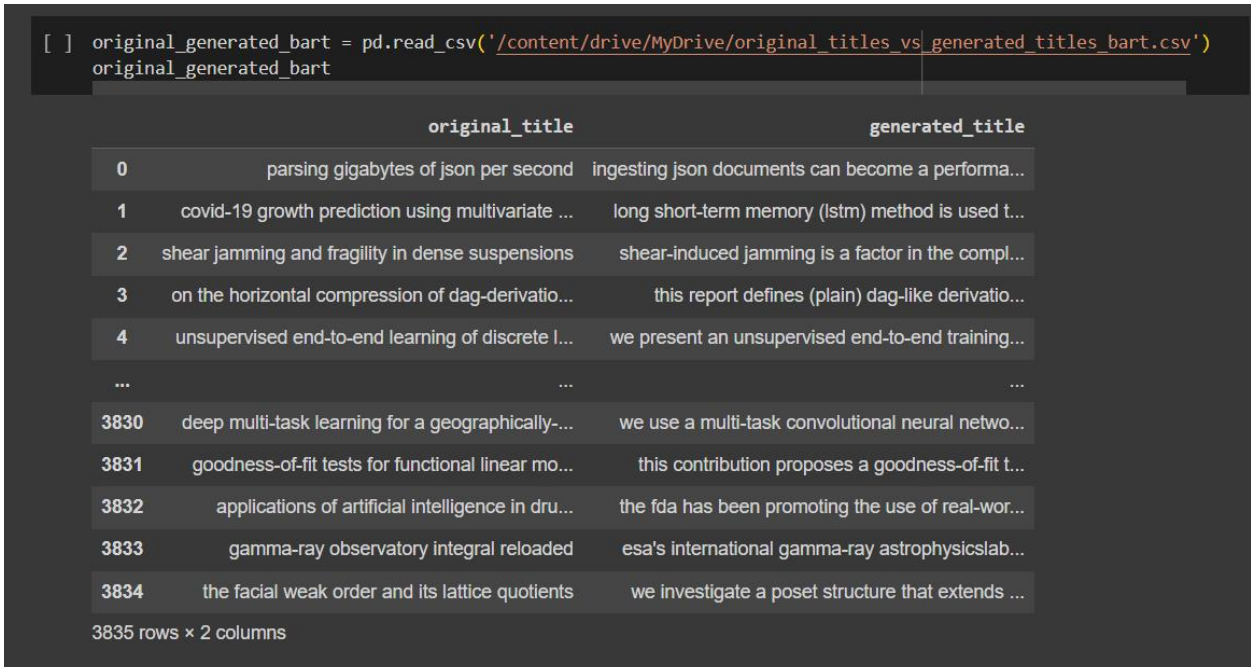
Figure 13: Evaluation of the Support Vector Machine (SVM) model.



```
model_name = "facebook/bart-large-cnn"
summarizer = pipeline("summarization", model=model_name, tokenizer=model_name)
```

```
Downloading: 100%  1.55k/1.55k [00:00<00:00, 44.8kB/s]
Downloading: 100%  1.51G/1.51G [00:41<00:00, 55.2MB/s]
Downloading: 100%  878k/878k [00:00<00:00, 2.22MB/s]
Downloading: 100%  446k/446k [00:00<00:00, 717kB/s]
Downloading: 100%  1.29M/1.29M [00:00<00:00, 2.26MB/s]
```

Figure 14: Train of the BART model.

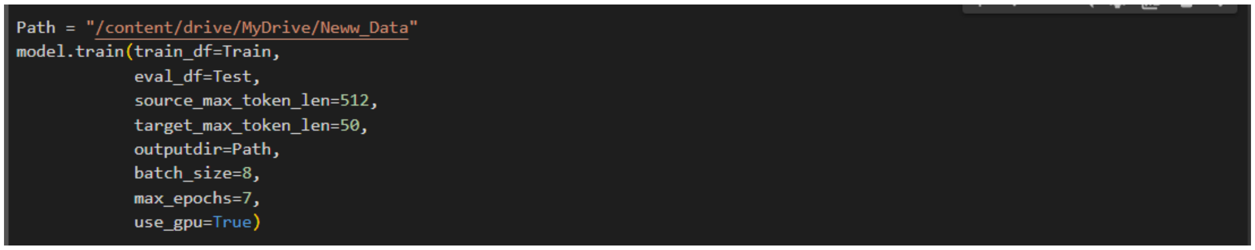


```
[ ] original_generated_bart = pd.read_csv('/content/drive/MyDrive/original_titles_vs_generated_titles_bart.csv')
    original_generated_bart
```

	original_title	generated_title
0	parsing gigabytes of json per second	ingesting json documents can become a performa...
1	covid-19 growth prediction using multivariate ...	long short-term memory (lstm) method is used t...
2	shear jamming and fragility in dense suspensions	shear-induced jamming is a factor in the compl...
3	on the horizontal compression of dag-derivatio...	this report defines (plain) dag-like derivatio...
4	unsupervised end-to-end learning of discrete l...	we present an unsupervised end-to-end training...
...	...	...
3830	deep multi-task learning for a geographically-...	we use a multi-task convolutional neural netwo...
3831	goodness-of-fit tests for functional linear mo...	this contribution proposes a goodness-of-fit t...
3832	applications of artificial intelligence in dru...	the fda has been promoting the use of real-wor...
3833	gamma-ray observatory integral reloaded	esa's international gamma-ray astrophysicslab...
3834	the facial weak order and its lattice quotients	we investigate a poset structure that extends ...

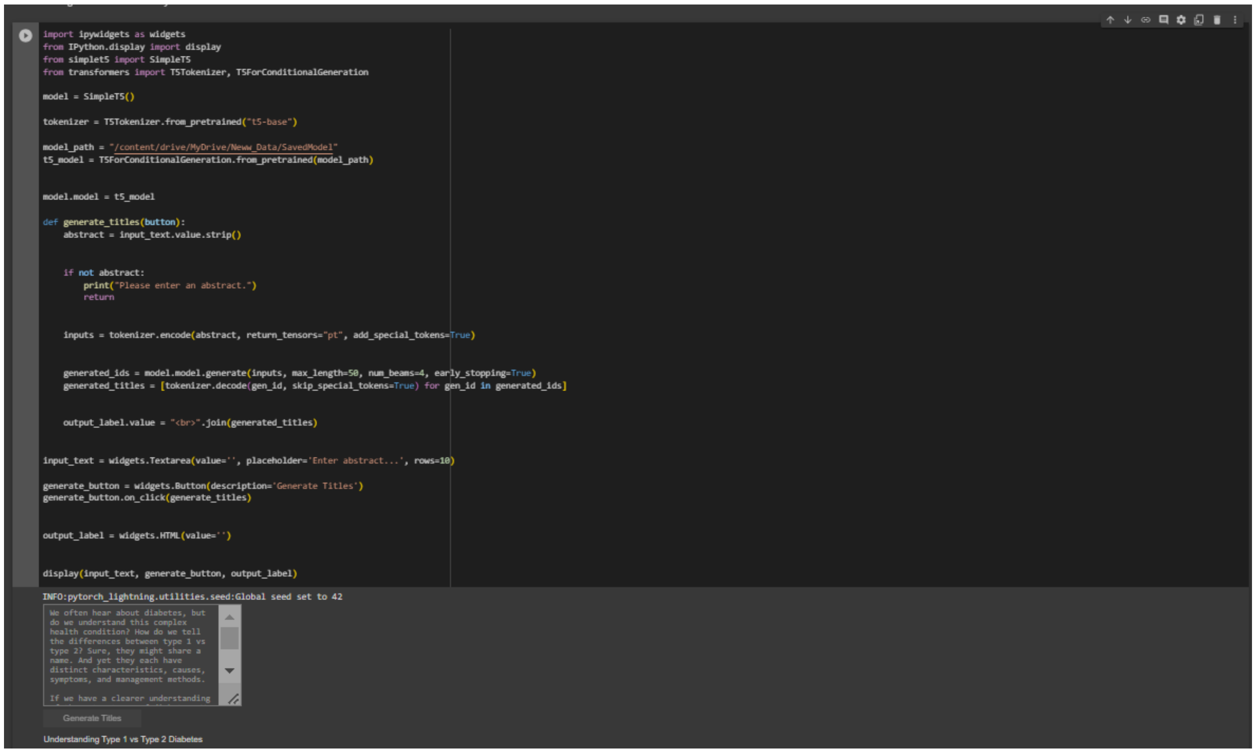
3835 rows × 2 columns

Figure 15: Result of the BART model.



```
Path = "/content/drive/MyDrive/Neww_Data"
model.train(train_df=Train,
            eval_df=Test,
            source_max_token_len=512,
            target_max_token_len=50,
            outputdir=Path,
            batch_size=8,
            max_epochs=7,
            use_gpu=True)
```

Figure 16: Train Simple T5 model.



```python
import ipywidgets as widgets
from IPython.display import display
from simplet5 import SimpleT5
from transformers import T5Tokenizer, T5ForConditionalGeneration

model = SimpleT5()

tokenizer = T5Tokenizer.from_pretrained("t5-base")

model_path = "/content/drive/MyDrive/Neww_Data/SavedModel"
t5_model = T5ForConditionalGeneration.from_pretrained(model_path)


model.model = t5_model

def generate_titles(button):
    abstract = input_text.value.strip()


    if not abstract:
        print("Please enter an abstract.")
        return


    inputs = tokenizer.encode(abstract, return_tensors="pt", add_special_tokens=True)


    generated_ids = model.model.generate(inputs, max_length=50, num_beams=4, early_stopping=True)
    generated_titles = [tokenizer.decode(gen_id, skip_special_tokens=True) for gen_id in generated_ids]


    output_label.value = "<br>".join(generated_titles)


input_text = widgets.Textarea(value='', placeholder='Enter abstract...', rows=10)

generate_button = widgets.Button(description='Generate Titles')
generate_button.on_click(generate_titles)


output_label = widgets.HTML(value='')


display(input_text, generate_button, output_label)
```

```
INFO:pytorch_lightning.utilities.seed:Global seed set to 42
```

```
We often hear about diabetes, but
do we understand this complex
health condition? How do we tell
the differences between type 1 vs
type 2? Sure, they might share a
name. And yet they each have
distinct characteristics, causes,
symptoms, and management methods.

If we have a clearer understanding
```

Understanding Type 1 vs Type 2 Diabetes

Figure 17: Result of Simple T5 model.

Chapter 5: Results and Evaluation

5.1 Testing Methodology

To estimate our automatic generating titles from abstracts, numerous experiments and models have been used to make a full study about this task and trying to reach to the most promising results. First, For T5 model, the text generation model and the assessment metrics appear to be tested in the code together with the remaining components of the system. It uses the model to produce titles, and then applies cosine similarity, BLEU, and Rouge to compare them to the original titles. A validation or training-testing separation is not specifically mentioned in the code. However, it divides the data using the Split function with a split data into 80% for training and 20% for testing. In this code, evaluation is conducted using the Test dataset. The T5 model appears to generate titles one by one in a loop. Second, For BART model, With the help of the provided code, the entire system is tested, including data preprocessing, title generation using the BART model, and evaluation using Rouge scores, BLEU scores, and cosine similarity. For the purpose of evaluating the system's overall performance, each component is integrated and tested in concert. The train_test_split function from scikit-learn is used in the code to separate training from testing. The data is divided into training and testing sets, with data_train getting 80% of the data and data_test getting 20%. After that, based on the testing set, multiple metrics are used to compare the created titles to the original titles. Instead of processing a batch of abstractions concurrently, the BART model generates titles one abstract at a time in a loop. Therefore, when creating titles from abstracts, the batch size is implicitly set to 1. Third, For SVM model, Data preparation, feature extraction with TF-IDF, training an SVM model, predicting titles for the test data, and assessing the generated titles with different metrics are all included. To gauge the system's overall performance, all of these parts are combined and tested. Since the model was trained using the SVM algorithm, which doesn't have a direct loss function like neural networks, loss is not important in this situation. Here, performance indicators like as BLEU score, Rouge score, and cosine similarity are used to assess the generated titles. The train_test_split function from scikit-learn is used in the code to separate training from testing. It divides the data into training and testing sets, allocating 20% of the data for testing and 80% of the data for training (train_data). The training features and matching target labels are used to train the SVM model. SVM is a classical machine learning technique, therefore unlike neural networks, it doesn't use batch processing. The complete feature matrix is used to train and test the model, and predictions are made for the entire test dataset simultaneously. In summary, the code tests the automatic title generation from abstracts using the T5, BART, and SVM models. The components of each model are tested collectively, and several metrics are used for evaluation. The models can generate titles one at a time or for the complete test dataset after dividing the data into training and testing sets.

5.2 Results

Numerous experiments have been implemented in our project, as was already showed.

5.2.1 Best Results Cases

In this case, the T5 model performs well, receiving respectably high scores for the Rouge-1, Rouge-2, and Rouge-L measures. The produced titles and the original titles have a significant amount of overlap, as seen by the F-scores, which range from 0.228 to 0.303. The scores for Precision and Recall are also respectably high. The produced titles and the original titles share a considerable amount of similarities, according to the average BLEU score of 0.102. The produced and original titles' semantic content appears to be very comparable, as indicated by the Average Cosine Similarity score of 0.421. Overall, this scenario shows that a T5 model can produce a successful result.

5.2.2 Acceptable Results Cases

In this case, the BART model yields outcomes that are deemed acceptable but not nearly as strong as the ideal outcome. There is some overlap between the produced titles and the actual titles, as indicated by the Rouge-1 and Rouge-L F-scores of 0.204 and 0.228, respectively. The Rouge-2 F-score of 0.086, however, indicates to a slightly lower percentage of overlap for bigram combinations. While the Average Cosine Similarity score of 0.260 indicates a moderate semantic similarity, the Average BLEU score of 0.027 indicates a moderate amount of similarity. While there is space for improvement, this scenario shows a respectable performance with the BART model overall.

5.2.3 Worst Results Cases

The SVM model performs poorly in this case when compared to the other models. The F-scores for Rouge-1 and Rouge-L are also relatively low, at 0.101 and 0.109, respectively. At 0.028, the Rouge-2 F-score is exceptionally low, indicating little bigram overlap. The Average Cosine Similarity score of 0.092 and the Average BLEU score of 0.008 both point to weak semantic similarity between the generated titles and the original titles. Overall, the three models performed the worst in this scenario, indicating the need for improvement or innovative approaches.

5.2.4 Limitations

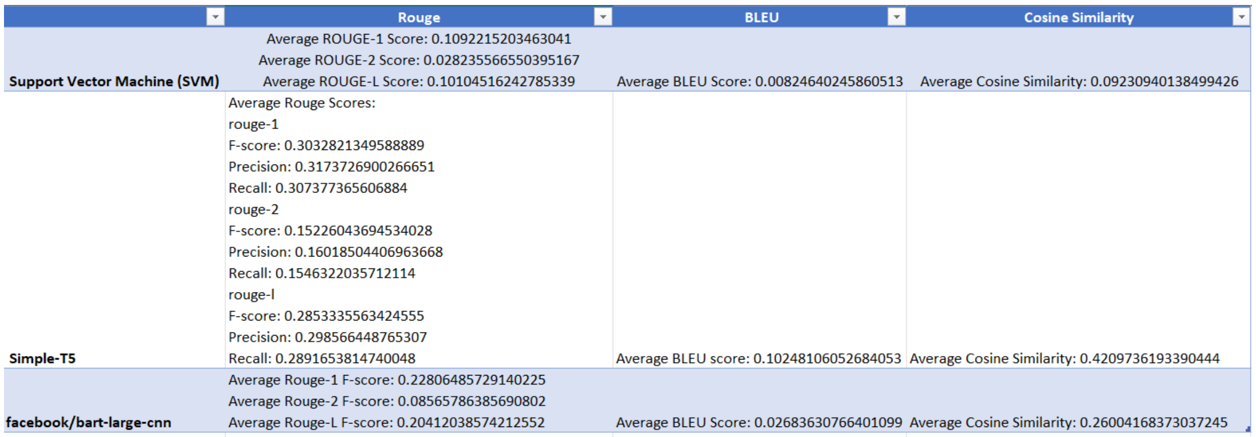
- First of all, there will be certain issues with integrating our concept into a real-world project. The first stage in data preparation is to collect and preprocess the data. A dataset with titles and abstracts is used in this case. To produce a relevant subset, the data is filtered according to specific requirements, such as the year of publication. Then, training and testing sets are created from the dataset. Different models are used, according to Model Training, to generate titles from abstracts. This example uses a number of models, including SimpleT5, BART, and SVM. Models are loaded and trained using the training dataset after the necessary packages and libraries are installed.

5.3 Evaluation

To evaluate the models' performance, their generated titles are contrasted with the original titles. The quality and similarity of the generated and original titles have been evaluated using evaluation metrics like ROUGE, BLEU, and cosine similarity.

5.3.1 Accuracy Evaluation

Table 5: Accuracy of Models.



	Rouge	BLEU	Cosine Similarity
Support Vector Machine (SVM)	Average ROUGE-1 Score: 0.1092215203463041 Average ROUGE-2 Score: 0.028235566550395167 Average ROUGE-L Score: 0.10104516242785339	Average BLEU Score: 0.00824640245860513	Average Cosine Similarity: 0.09230940138499426
Simple-T5	Average Rouge Scores: rouge-1 F-score: 0.3032821349588889 Precision: 0.3173726900266651 Recall: 0.307377365606884 rouge-2 F-score: 0.15226043694534028 Precision: 0.16018504406963668 Recall: 0.1546322035712114 rouge-l F-score: 0.2853335563424555 Precision: 0.298566448765307 Recall: 0.2891653814740048	Average BLEU score: 0.10248106052684053	Average Cosine Similarity: 0.4209736193390444
facebook/bart-large-cnn	Average Rouge-1 F-score: 0.22806485729140225 Average Rouge-2 F-score: 0.08565786385690802 Average Rouge-L F-score: 0.20412038574212552	Average BLEU Score: 0.02683630766401099	Average Cosine Similarity: 0.26004168373037245

A collection of measures called ROUGE is used to evaluate the accuracy of translations or summaries generated automatically by comparing them with reference summaries generated by humans. It measures the amount to which the generated and reference texts' n-grams (contiguous word sequences) overlap. ROUGE provides scores like ROUGE-1, ROUGE-2, and ROUGE-L that, respectively, focus on unigrams, bigrams, and the longest common subsequence. Better similarity between the reference and the generated texts is shown by higher ROUGE scores.

BLEU is a statistic frequently used for evaluating the accuracy of translations produced by automated means. In comparison to one or more reference translations, it evaluates how accurately the created translation uses n-grams. When calculating an overall score, BLEU considers precision into account at various n-gram levels (generally up to 4-grams). A perfect score of 1.0 indicates a perfect match to the reference translations, with higher BLEU evaluations indicating higher quality translations.

How similar two text texts are to one another can be evaluated using the cosine similarity metric. Using methods like TF-IDF, text documents used for natural language processing are transformed into vectors where each element indicates the frequency or importance of a specific word. The cosine of the angle between these vectors is calculated by the cosine similarity test. The obtained value represents the degree of word usage similarity between the texts. Cosine similarity has a range of -1 to 1, with larger values indicating higher similarity.

5.3.2 Time Performance

- Regarding the time, In the simple T5 model, for each epoch, it takes from 40 to 45 mins to implement the whole 12 epochs so the whole time will be in range 480 to 540 mins. While for facebook/bart-large-cnn it takes from 510 to 540 mins. and finally, SVM model takes 12 mins and 45 seconds to generate titles from abstracts.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

In conclusion, the main objective of this study is to generate scientific article titles using machine learning approaches. The development of a machine learning model addressed the difficulty faced by researchers with developing interesting and attractive titles that effectively represent their research. The objective is to collect and preprocess a large number of scientific titles and papers, use deep learning to generate connections between the titles and content, experiment with various parameters and architectures, evaluate the model's performance, and identify areas for improvement. The aim of this research intends to help researchers, especially non-native English speakers, create short, interesting, and informative titles by automating the title generating process.

The generation of scientific paper titles is a difficult task that requires understanding of the terminology used in the field and the capacity to effectively describe the research. In order to address this problem, a number of techniques have been developed, including rule-based systems, statistical models, and neural networks. While statistical approaches employ machine learning to find patterns from a collection of scientific papers, rule-based systems encode semantic and syntactic patterns to suggest titles. Deep learning techniques are used by neural network-based methods to automatically recognize patterns and provide excellent titles.

In our project, we experimented with different methods for generating scientific paper titles. The T5 model provided the best results, scoring highly on cosine similarity evaluations with accuracy 42%. Though not as strong as the T5 model, the BART model achieved acceptable outcomes with accuracy 30% in Rouge-1 F-score. In comparison to the other models, the SVM model performed poorly with accuracy 10.9%. In terms of creating accurate and relevant titles for academic papers, there is potential for growth and a need for innovative techniques.

6.2 Problem Issues

6.2.1 Technical issues:

- Our Automated system needs a huge amount of computational power and time on the CPU to run our models. and this was solved by using our GPU in running these experiments.

6.2.2 Scientific issues:

- To generate titles as much as possible of abstracts for model training and validation, many abstracts can affect the batch size because more than 8 batch size caused a system out of resources which means crash. Also, Large number of epochs which more than 8 epochs caused a system out of resources which means crash.

6.3 Future Work

Future research can use advanced pre-trained models like GPT-4 or T5-XXL to increase the accuracy and quality of title generation from abstracts. These models may identify more complex patterns and semantic connections in the text data since they have larger architectures and more parameters. The generated titles can be made more accurate, contextually relevant, and informative by using these more deeply pre-trained models.

References

[1] Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. Science, 349(6245), 261-266.(1)

[2] Brockopp, D. Y. (1983). What is NLP?. The American Journal of Nursing, 83(7), 1012-1014.(2)

[3] Kim, J. W., Weistroffer, H. R., & Redmond, R. T. (1993). Expert systems for bond rating: a comparative analysis of statistical, rule-based and neural network systems. Expert systems, 10(3), 167-172(3).

[4] Marjanović, M., Kovačević, M., Bajat, B., & Voženílek, V. (2011). Landslide susceptibility assessment using SVM machine learning algorithm. Engineering Geology, 123(3), 225-234.

[5] Pienaar, S. W., & Malekian, R. (2019, August). Human activity recognition using LSTM-RNN deep neural network architecture. In 2019 IEEE 2nd wireless africa conference (WAC) (pp. 1-5). IEEE.

[6] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.(10)

[7] Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., ... & Zettlemoyer, L. (2020). Multilingual denoising pre-training for neural machine translation. Transactions of the Association for Computational Linguistics, 8, 726-742.

[8] Liberman, M., & Cieri, C. (1998). The creation, distribution and use of linguistic data: the case of the linguistic data consortium. In LREC (pp. 159-166).

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, pages 6000–6010.(4)

[10] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.(5)

[11] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In NAACL.(6)

[12] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.(7)

[13] Ankur P Parikh, Oscar Tackstr ¨ om, Dipanjan Das, and ¨ Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In EMNLP.(8)

[14] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In ICLR.(9)

[15] Chen, Y., & Eger, S. (2022). Transformers go for the LOLs: Generating (humourous) titles from scientific abstracts end-to-end. arXiv preprint arXiv:2212.10522.(11)

[16] Dominik Beese, Begüm Altunba¸s, Görkem Güzeler, and Steffen Eger. 2022.(12)

[17] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. Educational and psychological measurement, 20(1):37–46.(13)

[18] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. 2016. Learning to learn by gradient descent by gradient descent.(14)

[19] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020.(15)

[20] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.(16)

[21] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-totext transformer. Journal of Machine Learning Research, 21(140):1–67.(17)

[22] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019.(18)

[23] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. 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.(19)

[24] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017.(20)