University of Science and Technology at Zewail City

Communications and Information Engineering Program

CIE 553: Natural Language Processing

Project Report

Team Information

Asmaa Mohammed Ibrahim	ID: 201701056
Elsayed Mohammed Elsayed Mostafa	ID: 201700316
Muhammed Magdy Alasmar	ID: 201700038
Salma Hasan Elbess	ID: 201601152

Table of Contents

Problem definition and motivation 3

Dataset 3

Objectives 3

Methodology 4

- Extractive Summarization 4
  - TF-IDF based summarization 4
  - TextRank based summarization 4
- Abstractive Summarization 5
  - PEGASUS 5
  - BART 5
  - Roberta2Roberta 6
  - SEQ2SEQ summarization 6

Results and Discussion 9

- Extractive Summarization 9
  - TF-IDF based summarization 9
  - TextRank based summarization 9
- Abstractive Summarization 9
  - Pretrained models - Transfer learning 10
  - SEQ2SEQ Model 11

Project Files 12

Individuals Contributions 12

References 12

Problem definition and motivation

Due to the increasing pace of life requirements in the world in parallel with the accelerating events around the world of conflicts and alliances, it has become difficult for many people to keep track of events around the world. On the other hand, following world events is an inherent right and even a duty of every human being to provide him with a general culture of the world around him and to be a thinking person with an opinion. Technologically, artificial intelligence has become an integral part of global development and production. Artificial intelligence has become an active and positive contribution in many different areas, most of the time. Therefore, it is also necessary for artificial intelligence to play a role in facilitating the access of information to humans around the world.

Dataset

In this project, we used the News Summary dataset available on Kaggle website [1]. This dataset has 4515 of news articles scraped from inshorts website along with their summaries. The columns exist in this dataset are Author, Date, Web Address, Headline, Summary and Complete Text. The headline and summary columns are human generated texts. For this project, we are only using the summary, which is usually 4 or 5 sentences to extract the headline in case of abstractive summarization and to get the most representative sentence in it with extractive summarization.

Objectives

The objective of this project is to have an efficient tool that can summarize any given article (sentences) into a specific predefined number of sentences in a way that is meaningful and readable. To do so, we are using the two main summarization approaches: extractive and abstractive summarization. In addition, the results of each approach is somehow compared to a related ground-truth summaries according to its type. The comparison metric is rouge which is a n-gram metric used for similarities.

Methodology

Extractive Summarization

For the extractive summarization, the main question is how to measure the sentence importance to know which sentences are selected as a representative summarization for the text. A lot of research has been conducted on this topic. Within the available approaches to measure the sentence's importance for summarization, we selected two algorithms to implement. These algorithms are TF-IDF based summarization and TextRank summarization.

TF-IDF based summarization

In this approach, the sentence importance is simply measured as the summation of the importance of its own words. For each word, the importance is the product of two terms namely TF and IDF. The term frequency (TF) is the frequency of the word within the sentence which is the number of word appearances in the sentence. While inverse document frequency (IDF) is the measure of how popular the word is within the whole document usually on a log scale. The output product is representing the word importance, hence the sentence importance is measured afterwards.

TextRank based summarization

TextRank is a graph based algorithm. It's roots go deep to the PageRank algorithm developed by Google. It was named PageRank after Larry Page, his algorithm was used in ranking web pages for online search results. The ranking algorithm is based on the assumption that the rank of a web page (W) is dependent on how important it is to other web pages. Suppose that a web page (X) is linked to another ten pages, then the contribution of (X) in the rank of (W) is the page rank of (X) divided by the total unique links it has. The same methodology applies to any other pages that may be related to (W). Then, by building the similarity matrix of N*N pages to be ranked. In the TextRank algorithm, web pages are represented by text sentences. Similarity matrix values represent the cosine similarity for example.Firstly, all sentences are initialized with a rank of 1.Then, the rank of a given sentence is calculated by summing all other pages' ranks weighted by their similarity values. After updating ranks for each sentence, they are arranged in a descending order. Then, the desired number of sentences is selected from top to bottom.[2]

Abstractive Summarization

For the abstractive summarization, it is aimed to generate a concise summary that captures the main ideas in an input text. The generated summary may contain new phrases that may not appear before in the input text. It is quite a challenging task as at first glance it requires humane characteristics, as abstraction requires some basic understanding of the source text content.

PEGASUS

BEGASUS was developed by google and improved the state of the art results gained from abstractive summarization. The ability for PEGASUS to deal with limited input examples. A thousand of data examples would be sufficient to be trained on and gaining good results. PEGASUS uses an encoder-decoder model for sequence to sequence learning. First, the encoder takes into account the whole input text then results in a context vector which could be considered as a numerical representation for the input text. Then, this context vector would be fed into the decoder whose mission is to translate this context vector into a summary. That encoder-decoder structure is called the transformer architecture.

BART

BART developed by Facebook AI. It is a denoising autoencoder. The model was trained by an input of corrupted sequences and the model's job is to remove the noise and extract the original sequence. The model can be used in many applications is natural language processing.[3]

Roberta2Roberta

This model is initialized on Roberta large. Roberta to Roberta model was tuned on the BBC XSum dataset [4] which is a news data set used for summarization training. As the objective of this project is news summarization we found that this model may be suitable and produce good results.

Hugging Face was used to get the three pretrained models in the code.[5]

SEQ2SEQ summarization

Sequence to sequence modelling is the basis of all summarization abstractive solutions, in which an encoder decoder model is formed using LSTMs.
This approach was not very promising, because we knew that starting a model from scratch will require intensive optimization and experimentation but we decided to go on with it to have hands-on-experience with basic abstractive models.

First step was to apply data processing and change sentences to sequences, and pad them, because the model will take vectors of numerical sequences of fixed size.
The process went as follows:

1. For the text examples we:
    a. Removed unnecessary spacing.
    b. Changed characters to lowercase
    c. Removed Unwanted punctuation
    d. Removed Stop words
2. For test samples:
    a. Same processing was done without removing stop words
    b. Start and end tokens were added
3. We used Keras Tokenizer to tokenize and change texts to sequences
4. We padded sequences to have a fixed sized input

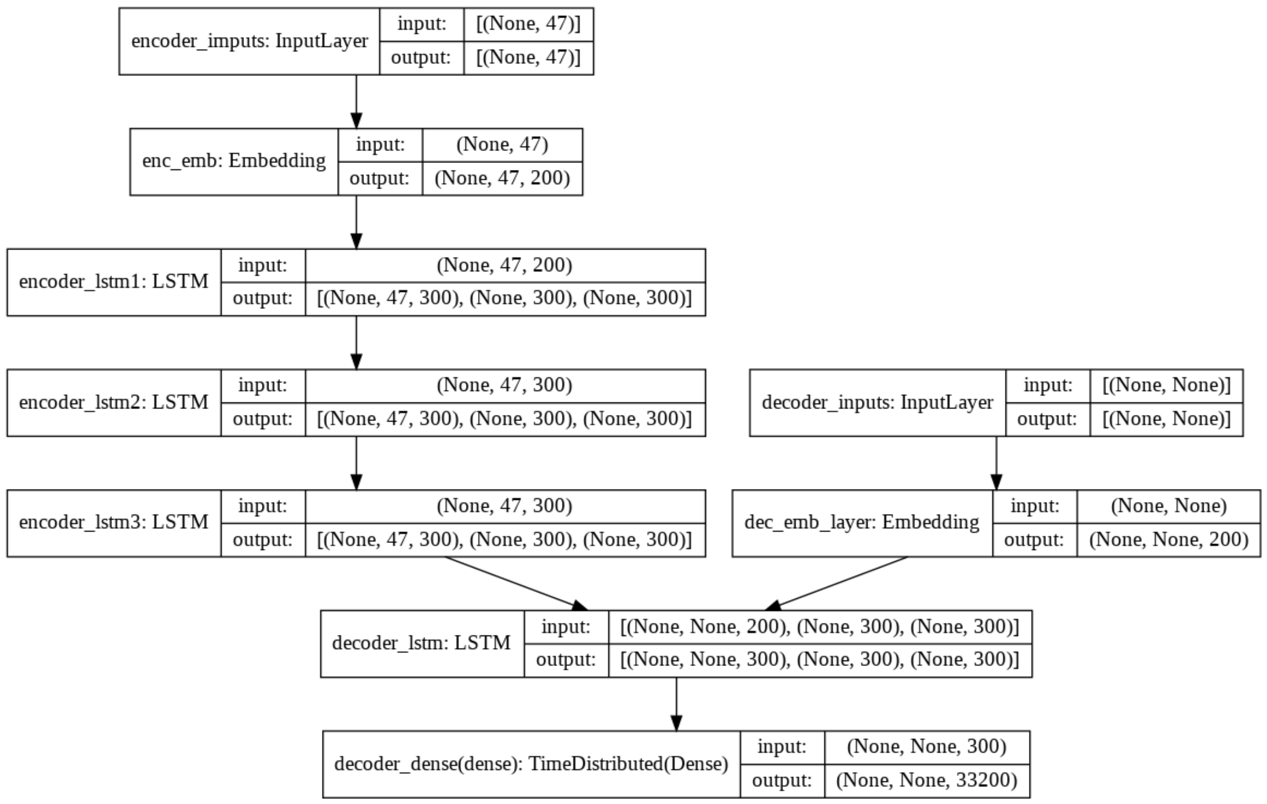
The second step was to create the encoder-decoder model, and train it on the dataset:
LAYERS:

Encoder:

- Encoder Embedding Layer
- 3 Encoder LSTM Layers

Decoder:

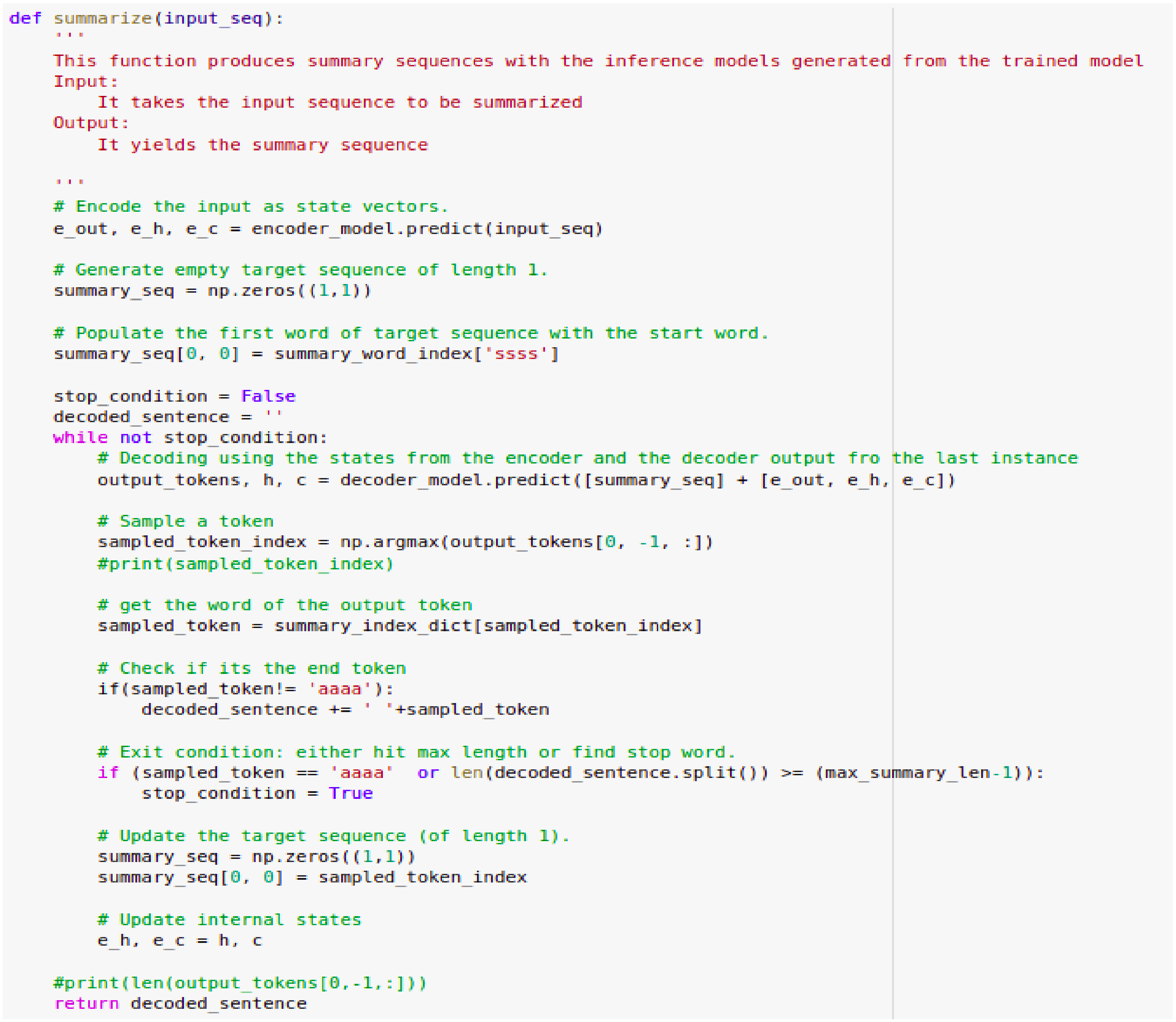
- Embedding Layer
- 1 LSTM Layer
- Dense Layer


encoder_imputs: InputLayer | input: [(None, 47)] | output: [(None, 47)]
enc_emb: Embedding | input: (None, 47) | output: (None, 47, 200)
encoder_lstm1: LSTM | input: (None, 47, 200) | output: [(None, 47, 300), (None, 300), (None, 300)]
encoder_lstm2: LSTM | input: (None, 47, 300) | output: [(None, 47, 300), (None, 300), (None, 300)]
encoder_lstm3: LSTM | input: (None, 47, 300) | output: [(None, 47, 300), (None, 300), (None, 300)]
decoder_inputs: InputLayer | input: [(None, None)] | output: [(None, None)]
dec_emb_layer: Embedding | input: (None, None) | output: (None, None, 200)
decoder_lstm: LSTM | input: [(None, None, 200), (None, 300), (None, 300)] | output: [(None, None, 300), (None, 300), (None, 300)]
decoder_dense(dense): TimeDistributed(Dense) | input: (None, None, 300) | output: (None, None, 33200)

After Training the model we had to divide it into two separate models:

1. Encoder_model: containing the encoding layers
2. Decoder_model: containing the decoding layers

This separation is because rnns generate each instance given the last ones. So the decoder needs to get the sequences decoded before as an input with each prediction.
We finally generated the summarization function. It code screenshotted below:



```
def summarize(input_seq):
    '''
    This function produces summary sequences with the inference models generated from the trained model
    Input:
        It takes the input sequence to be summarized
    Output:
        It yields the summary sequence

    '''
    # Encode the input as state vectors.
    e_out, e_h, e_c = encoder_model.predict(input_seq)

    # Generate empty target sequence of length 1.
    summary_seq = np.zeros((1,1))

    # Populate the first word of target sequence with the start word.
    summary_seq[0, 0] = summary_word_index['ssss']

    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:
        # Decoding using the states from the encoder and the decoder output fro the last instance
        output_tokens, h, c = decoder_model.predict([summary_seq] + [e_out, e_h, e_c])

        # Sample a token
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        #print(sampled_token_index)

        # get the word of the output token
        sampled_token = summary_index_dict[sampled_token_index]

        # Check if its the end token
        if(sampled_token!= 'aaaa'):
            decoded_sentence += ' '+sampled_token

        # Exit condition: either hit max length or find stop word.
        if (sampled_token == 'aaaa'  or len(decoded_sentence.split()) >= (max_summary_len-1)):
            stop_condition = True

        # Update the target sequence (of length 1).
        summary_seq = np.zeros((1,1))
        summary_seq[0, 0] = sampled_token_index

        # Update internal states
        e_h, e_c = h, c

    #print(len(output_tokens[0,-1,:]))
    return decoded_sentence
```

The results we got from our dataset were not very good, despite the optimization we had to do and the many learning epochs.
The results are shown in the next section.
To investigate if the small dataset was the main contributor to the failure of our model we read a bigger dataset 8000 instances and added it to our original dataset.
The results are below.

Results and Discussion

Extractive Summarization

In this section we are going to present the results of each approach.

TF-IDF based summarization

Using TF-IDF based summarization approach, we achieved relatively high accuracy with high efficiency as computation wise. To compare our results with a base model, we compared the output with BERT as a popular deep learning model (BERTSUM) for extractive summarization. Rouge metric was used and its scores were averaged along the whole text. The obtained results 73.5%, 83.68% and 68.6% as F1-score, precision and recall respectively. However, the actual accuracy is even higher as by comparing the two outputs, it was discovered that the only difference is due to the difference between the preprocessing output of our TF-IDF implementation and the BERTSUM model. For the computation efficiency, our implementation is a way better computationally than BERTSUM as our implementation spends only around 25 seconds on average to summarize the whole 3611 sentences. On the other hand, the BERTSUM model takse more than 90 minutes to do the same task.

TextRank based summarization

Using the TextRank based summarization approach, we achieved relatively good results compared with the TF-IDF approach discussed before. To compare our results with the TF-IDF approach, we tested 250 sentences only to discover that the TextRank approach takes around 314 seconds to summarize 250 sentences while as mentioned before the TF-IDF approach summarizes the whole 3611 sentences in about 25 seconds. This huge difference is due to the use of cosine similarity and word2vec modules and ready-made functions which probably takes a long time, especially the word2vec module.

Abstractive Summarization

In this section we are going to present the results of each approach we used for abstractive summarization.

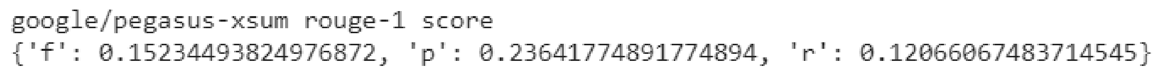
Pretrained models - Transfer learning

The models are tested using a sample test and Rouge-1 score

Sample test result for the three models using the same text.
Original Text*:* The new Nokia 3310 has a 2.4-inch colour screen unlike the original phone, which came with an 84x84 black and white display. The 2017 Nokia 3310's 1200mAh battery life is 10 times more than the original. The new model, which is slimmer and lighter than the original, has an updated version of the 'Snake' game, which was made by Gameloft
Original headline: How is the new Nokia 3310 different from its older version?
Headlines generated by the pretrained models
PEGASUS : Nokia has released a new version of its 3310 mobile phone.
BART: The new Nokia 3310 has a 2.4-inch colour screen unlike the original
Roberta2Roberta: Nokia has revealed that its flagship mobile phone is back on sale in the UK and Ireland
All three models performed well on the test sample generating a headline that summarizes the paragraph well.

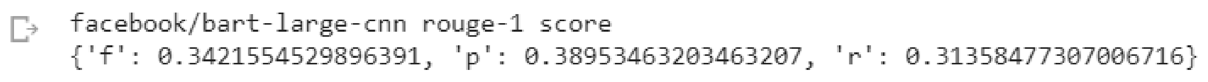
Rouge-1 results

Score is calculated on a small sample then taking the average.
PEGASUS



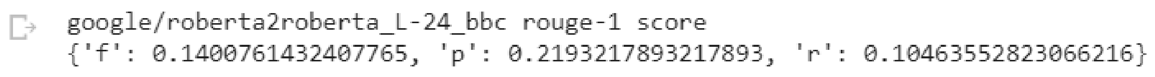
```
google/pegasus-xsum rouge-1 score
{'f': 0.15234493824976872, 'p': 0.23641774891774894, 'r': 0.12066067483714545}
```

BART



```
facebook/bart-large-cnn rouge-1 score
{'f': 0.3421554529896391, 'p': 0.38953463203463207, 'r': 0.31358477307006716}
```

Roberta2Roberta



```
google/roberta2roberta_L-24_bbc rouge-1 score
{'f': 0.1400761432407765, 'p': 0.2193217893217893, 'r': 0.10463552823066216}
```

The Rouge results were expected to be worse for the abstractive models. The abstractive model understands the context rather than extracting a representative sentence. This means that a headline might be different from the original one but still is very representative. In addition to that, we've used a pretrained model, which we did not train on our dataset, so the results are no way expected to be the same as the given headlines, but still the models are very efficient because with human judgement they are very representative.

SEQ2SEQ Model

With few training epochs the model tended to generate only stopwords, we attributed this to the abundance of those words. After further training the model would diverge to two sentences or three for any given headline as follows

```
Original Text:
The Kerala government on Tuesday said the state's tourism sector
registered a loss of about ?1,000 crore post the demonetisation drive by
the Centre. The number of foreign tourists visiting the state decreased by
10-15% while the number of domestic tourists dipped by 20-30%. The fall is
in contrast to the increase in the numbers before demonetisation, the
government added.


Generated headline:
I am blessed that you even know i exist ranveer to amitabh


Other Generated headlines:
Delhi metro to launch housing scheme with 2bhk 3bhk flats
```

We thought that the results should be due to the bias of the dataset and its small size. We added an extension to the dataset created by the same person in the dataset link. The new dataset is 80000 instances.
However, the model had the same behavior as the last one.
Which means that alternation of the base structure should be the way to deal with this network to get better results.

```
Original Text:
The Kerala government on Tuesday said the state's tourism sector
registered a loss of about ?1,000 crore post the demonetisation drive by
the Centre. The number of foreign tourists visiting the state decreased by
10-15% while the number of domestic tourists dipped by 20-30%. The fall is
in contrast to the increase in the numbers before demonetisation, the
government added.


Generated headline:
india s first lingerie vending machine to be brought down

Other headlines:
i am not a feminist i am a feminist i am scared karan johar
```

Project Files

Along with this report, the submitted project files contains the YouTube link for the video presentation, the presentation slides in PDF format and four notebooks as follows:

- *(Extractive_Summarization.ipynb):* for TF-IDF implementation and testing.
- *(Text_Rank_Extractive.ipynb):* for TextRank implementation and testing.
- *(SEQ2SEQ.ipynb):* for seq2seq abstractive model.
- *(Pretrained_abstractive_models.ipynb):* for the pretrained models and testing them.

Individuals Contributions

As a team, the workload was equally distributed between the team members. Although the fact that each one was assigned to a specific task, continuous supervision and feedback were given from each team member for each task. Here is a summary of the workload distribution.

- Dataset preprocessing was done by ***Salma Elbess.***
- TF-IDF extractive summarization by ***Elsayed Mostafa.***
- TextRank extractive summarization by ***Muhammed Alasmar***.
- Seq2Seq Abstractive summarization by ***Asmaa M. Ibrahim***.
- Pretrained abstractive summarization models by ***Salma Elbess***.

References

[1] Kondalarao Vonteru, *News Summary Dataset*. Available: https://www.kaggle.com/sunnysai12345/news-summary [Accessed 7 May 2021].
[2] Yadav, A., Kumar, M. and Pathre, A., 2020. Implemented Text Rank based Automatic Text Summarization using Keyword Extraction. *International Research Journal of Innovations in Engineering and Technology*, 04(11), pp.20-25.
[3]"BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension | Facebook AI Research", Ai.facebook.com, 2021. [Online]. Available: https://ai.facebook.com/research/publications/bart-denoising-sequence-to-sequence-pre-training-for-natural-language-generation-translation-and-comprehension/. [Accessed: 17- Jun- 2021].

[4]"Papers with Code - BBC XSum Benchmark (Text Summarization)", Paperswithcode.com, 2021. [Online]. Available: https://paperswithcode.com/sota/text-summarization-on-bbc-xsum. [Accessed: 17- Jun- 2021].
[5]"Hugging Face – The AI community building the future.", Huggingface.co, 2021. [Online]. Available: https://huggingface.co/models?filter=en&pipeline_tag=summarization. [Accessed: 17- Jun- 2021].