



**İstanbul  
Bilgi Üniversitesi**

**TITLE: COMPARISON OF THE TEXT  
SUMMARIZATION ALGORITHMS**

*by*

TUĞRA BURAK ÇAKICI, 117200073  
ARDA CEM ÖZMEN, 115200046

*Supervised by*

UZAY ÇETİN

*Submitted to the*

Faculty of Engineering and Natural Sciences  
*in partial fulfillment of the requirements for the*

Bachelor of Science

*in the*

Department of Computer Engineering

Jan, 2021

## ***Abstract***

*Text summarization is accepted as a hard task by NLP community. Availability of data structures for text summarization is quite hard and there is a known difficulty to their production. This project's aim is to create a general text summarization capitalizing from the knowledge of LSTM, RNN and many other libraries and methods that have text summary as one of their objective. By learning their limits and their optimum situation, a product powered by FastAPI will be developed to serve actively to people..*

# TABLE OF CONTENTS

<b>Abstract</b>	<b>ii</b>
<b>Table of Contents</b>	<b>iii</b>
<b>List of Figures</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Pre Information</b>	<b>1</b>
2.1 Natural Language Processing . . . . .	1
2.2 General Information Regarding Coding Progress . . . . .	1
2.3 Data Preperation . . . . .	2
2.4 RNN . . . . .	2
2.5 LSTM . . . . .	3
<b>3 Text Summarizaton Methods</b>	<b>5</b>
3.1 Extractive Summarization Methods . . . . .	6
3.2 Abstractive Summarization Methods . . . . .	6
<b>4 Textrank</b>	<b>7</b>
4.1 Work Principle of TextRank . . . . .	7
4.2 An Example of TextRank Text Summarization . . . . .	7
<b>5 Sequence2Sequence</b>	<b>12</b>
5.1 An Example of Seq2Seq Text Summarization . . . . .	12
5.2 The Difficulties and Variables of Seq2Seq . . . . .	16
5.3 Its Purpose In Text Summarization . . . . .	16
<b>6 Named Entity Recognition(NER)</b>	<b>17</b>
<b>7 Transformers</b>	<b>17</b>
<b>8 Discussion</b>	<b>19</b>
8.1 Current Results That Have Concluded . . . . .	19
8.2 Risk Analysis . . . . .	20
<b>9 Conclusion</b>	<b>20</b>
<b>References</b>	<b>21</b>

## LIST OF FIGURES

1	An unrolled RNN . . . . .	2
2	A Simple RNN . . . . .	3
3	LSTM module . . . . .	4
4	LSTM cell state . . . . .	4
5	LSTM forget layer . . . . .	5
6	LSTM input gate . . . . .	6
7	LSTM output layer . . . . .	6
8	Similarity Matrix . . . . .	9
9	A basic Seq2Seq model . . . . .	12
10	Data header . . . . .	13
11	Result of a text summary using Seq2Seq . . . . .	16

# **1 Introduction**

As the time goes on and technology refines itself, the daily lives of people are getting much more easier thanks to said advancements. Although some can point out that some functions that are done by machine now can lead to some tardiness in people's part by not doing them, it is, ultimately, a step in forward. There are many actions that are taken care of an some sort of AI now, one of them is text summarization. This project's main aim is to understand how can text summarization made by deep learning can be ideal for everyone, ultimately making them gain some time as they go along. As long as a product that is fast, precise and practical will be made to serve people, this project will be a success.

## **2 Pre Information**

There are some parts that are need to be known before delving into the methods.

### **2.1 Natural Language Processing**

NLP is a subsection of Artificial Intelligence. Within the software development and any computer engineering related objectives, there are two different types of languages that are used. They divide into programming languages and natural languages. The process of human language and demenaor translating into a programming language is described as NLP. NLP aims to see the balance between these two types of languages and try to integrate one to another in an elegant and practical way. The some areas that NLP is used are: Text classification and categorization, Named Entity Recognition, Part-of-speech tagging, Semantic Parsing and Question Answering, Paraphrase Detection and so on.

### **2.2 General Information Regarding Coding Progress**

As it is with every other deep learning an machine learning application, usage of Python was the smart choice because of the already built in libraries that can be counted on. Since this project delves in network models, the usage of keras library's default models and tensorflow's computation techniques, the construction of a simulation of a network system was quite smooth. Finally, networkx will be used to see the general behavior of the model that has been built.

The building process will be a bit detailed. To get the LSTM and other layers needed to built Seq2Seq Encoder-Decoder Model, keras will pull its default layers to be filled by. It can also help with tokenizer to tokenize the text data, which, in different parts, will help for model to understand the behaviour and importance of words and sentences within the text. Just like any other model, sklearn will be used to split text data to train and validation values. The other functions such as numpy and pandas will be included for obvious reasons.

## 2.3 Data Preperation

Within the experiments of TextRank, a text data that has been prepared by the author with 5 unique texts has been integrated into the kernel. With using beatifulsoup and pandas libraries, the embedding of data was quite easy. There wasn't much cleaning process for TextRank data. For Seq2Seq model, cleaning was done using spacy library.

## 2.4 RNN

Since text summarization heavily relies on every bit of information it can extract from the text, It heavily relies on the previous knowledge that the model gets from examining through the lines. This kind of solution proposes a reccurent template for the model, which most of the normal networks fail to comply. This is, in theory, where Recurrent Neural Network comes in to create a system that reccurently give information within its hidden states from the previous models. With their reccurent system, text summarization objective becomes much easier.

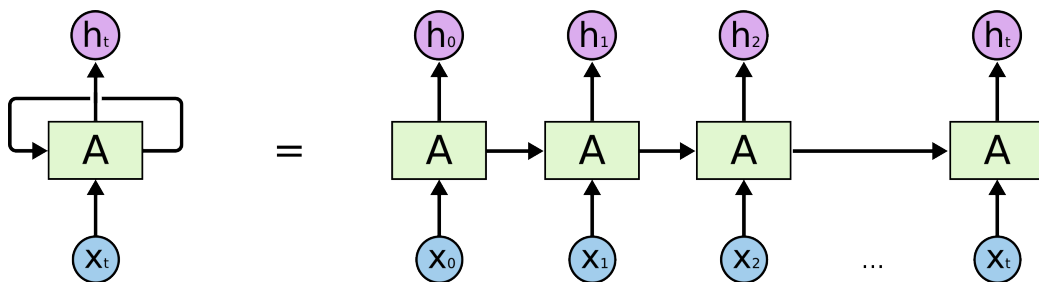


Figure 1: An unrolled RNN

RNN's are used in multiple different objectives such as peech recognition, language modeling, translation, image captioning and so on. Its capability to hold important information while proceeding to another network model

enables other networks to work with more data and get results according to the previous data. In terms of text summarization, whenever a text is going through a model, the bits that it picks up will help to build the general idea behind the text.

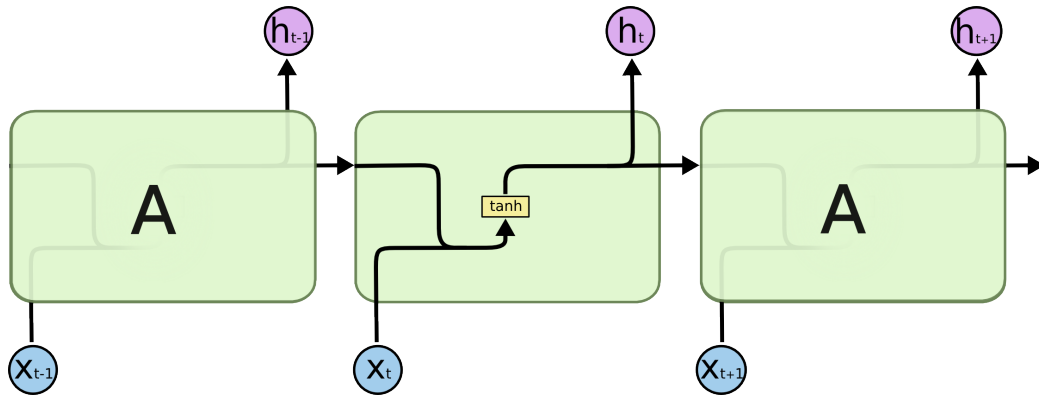


Figure 2: A Simple RNN

There is also the matter of attention layer, which helps the networks to focus on specific information about converted networks throughout the whole sequence. This has not been used yet for the sole reason of not understanding it enough to implement it manually. The general idea of attention layer was best understood in Madhav Mishra’s Seq2Seq: Abstractive Summarization Using LSTM and Attention Mechanism implementation[Mis]. Although there are default versions of attention layer provided by keras and such, one needs to understand every aspect of what they are building in order to get the best result.

Obviously, as it has stated before, RNN should work like a charm for text summarization. Unfortunately, it has a big negative going against it. While it can remember recent information quite well, it struggles to remember when the info is far away from the present model. This can be traced back to its recurrent form where it can only remember one specific information. It is explained as Vanishing Gradient Problem, where a model struggles to learn going forward because it starts to forget previous computations from former layers. For this, LSTM would be the most appropriate network model to use, which will be talked about.

## 2.5 LSTM

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They

were introduced by Hochreiter Schmidhuber (1997), and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems, and are now widely used.

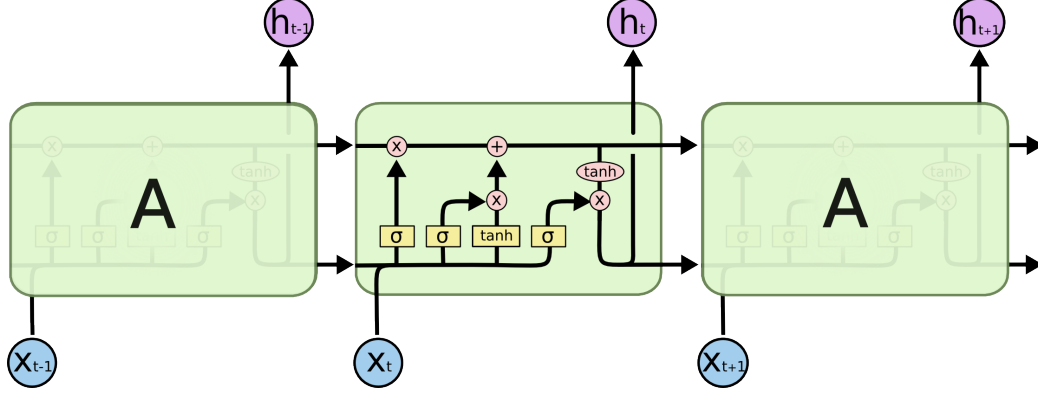


Figure 3: LSTM module

Since they are designed to avoid long-term dependencies, LSTM's are built different compared to normal RNN's. Although their blueprints are similar, their executive variables and procedures are quite different. They both share a chain like module, but the main difference is LSTM's include various gates and a state to make interaction between LSTM modules more easier. They are more capable to interact, remember and forget compared to normal RNN's. In a field such as text summarization, every information should be looked over once the text is completed, and LSTM becomes a reasonable tool to use.

The core part of the LSTM's are their cell state. The cell state, in theory, can carry relevant information throughout the processing of the sequence. So even information from the earlier time steps can make it's way to later time steps, reducing the effects of short-term memory.

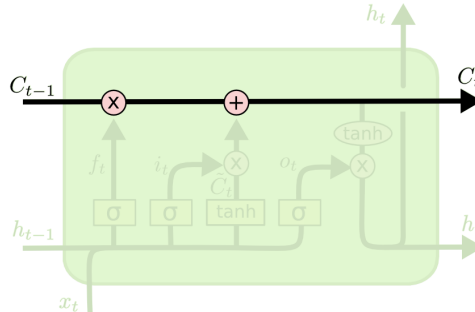
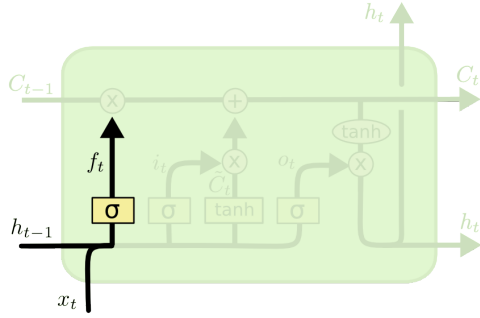


Figure 4: LSTM cell state



Next part is the forget gate layer, where a sigmoid decides whether to keep the information coming from the previous state or not. It will either give 1, i.e. keep, or 0, i.e. forget, and carry the decision to the cell state. This information can be about a label value that may become important in future. Text summarization wise, it can remember a part where model feels it can be important when starting the summarization process.



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Figure 5: LSTM forget layer

Next step will be consisting of two layers, the input gate layer and tanh layer. The input gate will pass the previous hidden state and current input into a sigmoid function. That decides which values will be updated by transforming the values to be between 0 and 1. 0 means not important, and 1 means important. The tanh layer creates a vector of new candidate values, that could be added to the state. By multiplying these values, input part will decide what to remember from tanh part.

The last step is the output gate layer, where the output will be based on the cell state, but will be a filtered version. Sigmoid will decide which parts that will be the output, then tanh will filter these values for the hidden state.

The visuals and main idea for this summary of RNN and LSTM modules have been taken from Christopher Olah's Understanding LSTM Networks piece[Ola]. For related work and more information, please take a look

### 3 Text Summarization Methods

With the progression of NLP techniques, work fields such as machine learning and deep learning upped the scale on what can a network model can do. As talked about before, this Project looks into these capabilities, but specifically text summarization methods. There are multiple ways to summarize a text using any number of algorithm, some will be talked about here, some won't because restriction issues. Text summarization methods themselves divide to two within itself.

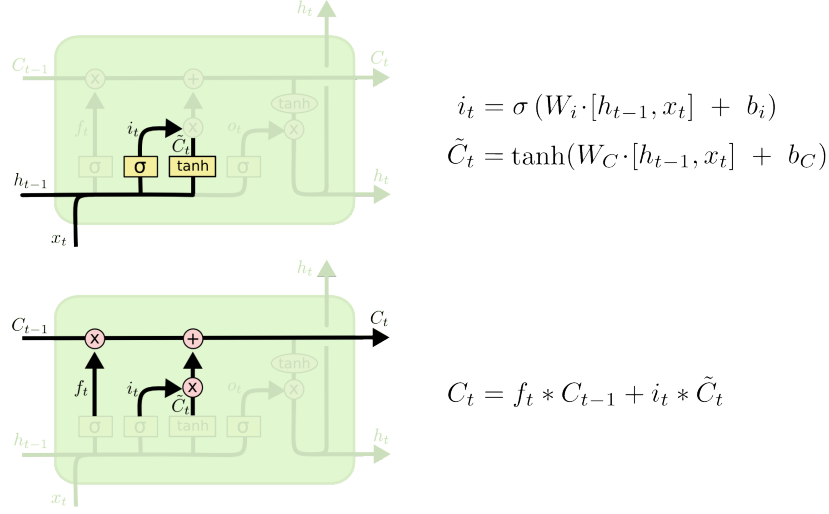


Figure 6: LSTM input gate

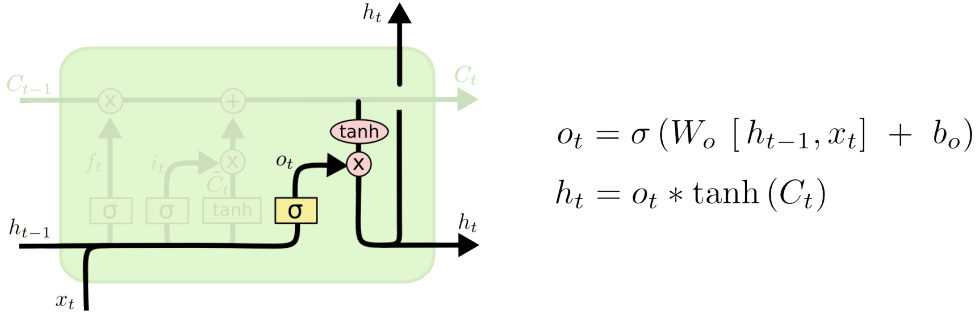


Figure 7: LSTM output layer

### 3.1 Extractive Summarization Methods

This type of summarization techniques extract the summary from the text itself, meaning a part of a text will be understood as something close to a summary of it, more generally its main idea. The most famous of these methods is TextRank, a method which uses the principles of PageRank, which will be talked about.

### 3.2 Abstractive Summarization Methods

Abstractive methods will try to come up with their own summary using models designed to value the information it gathers from all text body. The model will generate The summary in own words. The most used methods are Seq2Seq, Transformers library and different types of close source algorithms.

## 4 Textrank

Textrank is a text processing graph-based ranking model that can be used to identify the most important sentences in the text. TextRank's basic concept is to give a score to each sentence for their importance, then sort them accordingly. The first sentence that is shown is to be believed as the main idea of the text, also can be understood as its summary.

### 4.1 Work Principle of TextRank

The first step is to divide the text into sentences. After separating all the sentences in the text into a list, a graph is created with nodes, each sentence is a node, these nodes are linked by weight and similarity. These variables will be ranked with. This will be a stage where the most time is spent and where it will be thought to increase creativity. Different similarity scores created as a result of these improvements will significantly change the TextRank result. These similarity scores will be put into a graph that will help to decide which sentence is the most important. After getting the result of the graph, PageRank will be run. PageRank algorithm is developed for websites that cite each other. Whichever website is cited the most out of the given addresses will be seen as the root of those given websites. As an example for PageRank, suppose there is a node 'A'. If website node 'A' and its neighbors and weights are 'B' (0.65) 'C' (0.04) 'D' (0.27) then the probability of citing from website A to each of those nodes as:

$$A \text{ to } B = 0.65 / (0.65 + 0.04 + 0.27)$$

$$A \text{ to } C = 0.04 / (0.65 + 0.04 + 0.27)$$

$$A \text{ to } D = 0.27 / (0.65 + 0.04 + 0.27)$$

With 'A' probability being the highest, website 'A' will be considered as the main website that cited for all given website. This example will should work as an indication for TextRank, where instead of references, the words are counted and weighted. By choosing the sentence with the most importance rank, the summary, or as it is thought main idea, will be extracted.

### 4.2 An Example of TextRank Text Summarization

As it is with every other deep learning project, extracting the data from its original form is the first step. Cleaning should also be done to get the most accurate solution possible. The texts within the data will be split into sentences in order to making the ranking decision easier. After that, the words that have been used in those texts will be counted with the intention

of importance ranking. The sentence with the most used words will be a front runner.

```
1 from collections import Counter
2 def bag_of_words(sentence):
3     return Counter(word.lower().strip('.',') for word in sentence
4     .split(' '))
5     #This part will count the words given in a sentence. It will
6     be useful for PageRank part when the importance of words
7     gets evaluated.
8 bag_of_words(sentences[0])
9 >>> Counter({'the': 2,
10             'covid-19': 1,
11             'pandemic': 3,
12             'also': 1,
13             'known': 1,
14             'as': 1,
15             'coronavirus': 3,
16             'is': 1,
17             'an': 1,
18             'ongoing': 1,
19             'of': 1,
20             'disease': 1,
21             '2019': 1,
22             '(covid-19)': 1,
23             'caused': 1,
24             'by': 1,
25             'severe': 1,
26             'acute': 1,
27             'respiratory': 1,
28             '2': 1,
29             '(sars-cov-2)': 1})
```

Code 1: Word count for PageRank

The next part will be converting these sentences and all different words into vectors. After finding the similarity rates for every sentence according to the words that have been used, these rates will be put into a similarity matrix for each sentence that system processes.

```

Out[24]: array([[1.0, 0.04222127, 0.12365039, 0.11414665, 0.19566879,
0.0628034, 0.04794437, 0.04534055, 0.02517598, 0.
0.06184139, 0.05833562, 0.02906037, 0.0800811, 0.18038161,
0.04222127, 1.0, 0.17338549, 0.04991025, 0.
0.03695374, 0.0723026, 0.07133342,
0.03256068, 0.0, 0.0, 0.0, 0.0,
0.12365039, 0.17338549, 1.0, 0.0807391, 0.02313602,
0.0794774, 0.00983075, 0.0, 0.0431015, 0.10032384,
0.01897858, 0.03213329, 0.01232811, 0.07427931, 0.13489198,
0.07840652, 0.0, 0.01478899, 0.03068369,
0.11414665, 0.04991025, 0.0807391, 1.0, 0.12648709,
0.0, 0.02290888, 0.0, 0.01932959, 0.0,
0.01949534, 0.02450632, 0.07110186, 0.0,
0.04092856, 0.07946815, 0.0, 0.12220351,
0.19566879, 0.0, 0.02313602, 0.12648709, 1.0,
0.0623753, 0.0, 0.0, 0.0808679, 0.0,
0.08442843, 0.08156137, 0.0, 0.07020327, 0.0,
0.06534136, 0.0, 0.0, 0.06702297,
0.0628034, 0.0, 0.0794774, 0.0, 0.0623753,
1.0, 0.0, 0.0, 0.13585547, 0.11362795,
0.08231555, 0.0958959, 0.0, 0.05623581, 0.13356016,
0.03569441, 0.0, 0.07906706, 0.03084043,
0.04794437, 0.03695374, 0.00983075, 0.02290888, 0.0,
0.0, 1.0, 0.12101667, 0.05359797, 0.12286705,
0.01934708, 0.0, 0.01256748, 0.06506712, 0.01101358,
0.08610224, 0.07221752, 0.01507614, 0.097472,
0.04534055, 0.0723026, 0.0, 0.0,
0.0, 0.12101667, 1.0, 0.08244652, 0.05819646,
0.0, 0.0, 0.0, 0.08119824, 0.0,
0.04310106, 0.0, 0.0, 0.0,
0.02517598, 0.0, 0.0431015, 0.01932959, 0.0808679,
0.13585547, 0.05359797, 0.08244652, 1.0, 0.08483707,
0.0221233, 0.12100917, 0.01437085, 0.06773497, 0.0661342,
0.06318795, 0.0, 0.01723949, 0.04454876,
0.0, 0.07133342, 0.10032384, 0.0,
0.11362795, 0.12286705, 0.05819646, 0.08483707, 1.0,
0.04684846, 0.0, 0.03043182, 0.03368727, 0.05318782,
0.09448181, 0.05829097, 0.07280711, 0.08920658,
0.0, 0.0, 0.01897858, 0.0, 0.08442843,
0.08231555, 0.01934708, 0.0, 0.0221233, 0.04684846,
1.0, 0.0, 0.02426193, 0.02685735, 0.02126208,
0.05114128, 0.0, 0.02910498, 0.15652472,
0.05833562, 0.0, 0.03213329, 0.01949534, 0.08156137,
0.0958959, 0.0, 0.0, 0.12100917, 0.0,
0.0, 1.0, 0.0, 0.05227123, 0.05399934,
0.03317798, 0.0, 0.0, 0.0,
0.02906037, 0.0, 0.01232811, 0.02450632, 0.0,
0.0, 0.01256748, 0.0, 0.01437085, 0.03043182,
0.02426193, 0.0, 1.0, 0.05457211, 0.01381142,
0.06625337, 0.04528167, 0.01890601, 0.04486997,
0.0800811, 0.0, 0.07427931, 0.07110186, 0.07020327,
0.05623581, 0.06506712, 0.08119824, 0.06773497, 0.03368727,
0.02685735, 0.05227123, 0.05457211, 1.0, 0.08028592,
0.11614994, 0.13673823, 0.02092849, 0.0690271,

```

Figure 8: Similarity Matrix

These similarities will be put inside a network graph that will use PageRank algorithm to sort the similarities of these values. The closest to 1 will be the sentence with the most similarity, thus assuming it as the main idea.

```

1 import networkx as nx
2 nx_graph = nx.from_scipy_sparse_matrix(similarity_graph)
3 scores = nx.pagerank(nx_graph)
4 scores
5
6 >>> {0: 0.058607608354242294,
7      1: 0.04601850507134548,
8      2: 0.057010612605271425,
9      3: 0.05146310022030544,
10     4: 0.05292206864494559,
11     5: 0.055478147303211045,
12     6: 0.052324712879283315,
13     7: 0.04650125802396069,

```

```

14 8: 0.05514990276837194,
15 9: 0.05748267984597134,
16 10: 0.04760188651193677,
17 11: 0.046632574078683535,
18 12: 0.04481193684950849,
19 13: 0.060230270320510274,
20 14: 0.056320045974837064,
21 15: 0.061130538190084016,
22 16: 0.04689382572969491,
23 17: 0.04587183540789658,
24 18: 0.0575484912199396}

```

Code 2: PageRank results

At the end, there will be a function to sort the text with the sentences of most importance

```

1 print(textrank(document))
2 >>> [(0.061130538190084016,
3      'It has led to the postponement or cancellation of events ,
4      widespread supply shortages exacerbated by panic buying ,
5      agricultural disruption and food shortages , and decreased
6      emissions of pollutants and greenhouse gases. '),
7      (0.060230270320510274,
8      'Many places have also worked to increase testing capacity and
9      trace contacts of the infected. '),
10     (0.058607608354242294,
11     'The COVID-19 pandemic , also known as the coronavirus pandemic
12     , is an ongoing pandemic of coronavirus disease 2019 (COVID
13     -19) caused by severe acute respiratory syndrome coronavirus
14     2 (SARS-CoV-2). '),
15     (0.0575484912199396,
16     'There have been incidents of xenophobia and discrimination
17     against Chinese people and against those perceived as being
18     Chinese or as being from areas with high infection rates. '),
19     (0.05748267984597134,
20     'Recommended preventive measures include social distancing ,
21     wearing face masks in public , ventilation and air-filtering ,
22     hand washing , covering one's mouth when sneezing or coughing ,
23     disinfecting surfaces , and monitoring and self-isolation for
24     people exposed or symptomatic. '),
25     (0.057010612605271425,
26     'The World Health Organization declared the outbreak a Public
27     Health Emergency of International Concern in January 2020 and
28     a pandemic in March 2020. '),
29     (0.056320045974837064,
30     'The pandemic has caused global social and economic disruption
31     , including the largest global recession since the Great
32     Depression. '),
33     (0.055478147303211045,

```

```

17     'The virus spreads mainly through the air when people are near
18     each other. '),
18     (0.05514990276837194,
19     'People remain infectious for up to two weeks, and can spread
19     the virus even if they do not show symptoms. '),
20     (0.05292206864494559,
21     'Symptoms of COVID-19 are highly variable, ranging from none
21     to severe illness. '),
22     (0.052324712879283315,
23     '[b It leaves an infected person as they breathe, cough,
23     sneeze, or speak and enters another person via their mouth,
23     nose, or eyes. '),
24     (0.05146310022030544,
25     'As of 28 December 2020, more than 81.1 million cases have
25     been confirmed, with more than 1.77 million deaths attributed
25     to COVID-19. '),
26     (0.04760188651193677,
27     'Several vaccines are being developed and distributed. '),
28     (0.04689382572969491,
29     'Many educational institutions have been partially or fully
29     closed. '),
30     (0.046632574078683535,
31     'Current treatments focus on addressing symptoms while work is
31     underway to develop therapeutic drugs that inhibit the virus
31     . '),
32 ]

```

Code 3: TextRank Output

## 5 Sequence2Sequence

Sequence to Sequence (also known as seq2seq) is a bundle of approaches made for RNN's which delve in language translation, image captioning, conversational models and text summarization. Developed by Google, seq2seq is commonly known for playing the fundamental part on Google's speech recognizing apps and Google Translate. It can also work wonders with text summarization, which will be the aim of this Project.

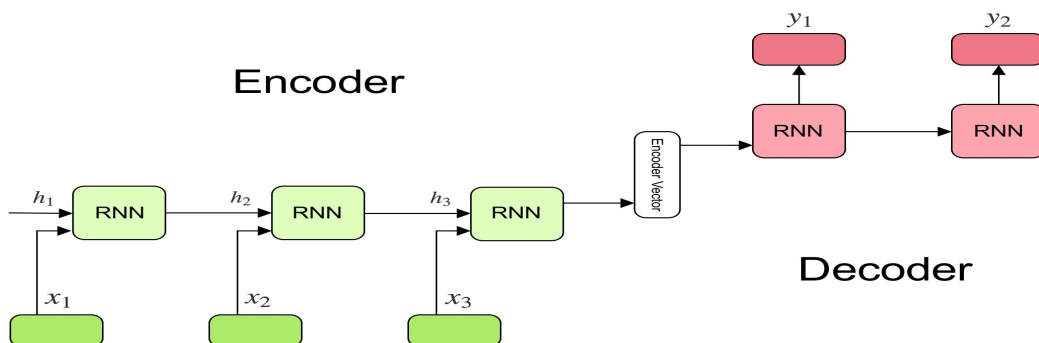


Figure 9: A basic Seq2Seq model

Since seq2seq frequently uses RNN's and LSTM's, it should be pretty obvious that the sequence model's present feeds from its past. It uses an encoder-decoder architecture that enables the model to go along within sequences efficiently while producing an end result because of it. Encoder reads the input sequence and summarizes the information as hidden state and cell vectors whereas decoder takes the input sequence from the final cell of the encoder and turns it into the output sequence.

In order to develop an encoder-decoder system for seq2seq, two network models needed to be built to compensate this requirement. Recurrent Neural Network can be considered as an acceptable tool but its inability to have long term dependencies causes summarization to be less accurate. For this purpose, the usage of the lstm can be seen as the right choice for it.

### 5.1 An Example of Seq2Seq Text Summarization

This example is taken from another source. If needed, the link of this example will be in references. A general review of an already built system seemed like the correct path to start. With generalizing some components to next steps of this project, reviewing how a seq2seq system would work in a system designed for text summarization will help to decide if it is feasible to use seq2seq algorithm in conditional ways. These variables will be taken



into consideration when this project's aimed application is built wondering if seq2seq can be used in that.

The data that will be used for this template is roughly 30 MB. It is not a huge data per se, but it is big enough for model to get a respectable loss value. The data will contain the original text and its headline. The headlines will act as the comparable summaries of the given texts within the model and its predicted summary value to be compared.

Out[5]:

	text	summary
0	Saurav Kant, an alumnus of upGrad and IIT-B's...	upGrad learner switches to career in ML & AI w...
1	Kunal Shah's credit card bill payment platform...	Delhi techie wins free food from Swiggy for on...

Figure 10: Data header

Since data itself has been taken from various websites, there are bound to be some parts that would cause problems meaning and understanding wise, so the appropriate action is to clean the data for it to be more friendly to the general language. Regular Expression library (re) will be used to briefly clean data. NLP library spacy will also be used to speed up the cleaning process by using its pipe function. Since the function has NER enabled (will be talked about) it needs to be turned off to speed up even more.

```

1 from time import time
2 import spacy
3 nlp = spacy.load('en', disable=['ner', 'parser']) # disabling
    Named Entity Recognition for speed
4
5 #Taking advantage of spaCy .pipe() method to speed-up the
    cleaning process:
6 #If data loss seems to be happening(i.e len(text) = 50 instead
    of 75 etc etc) in this cell , decrease the batch_size
    parametre
7
8 t = time()
9
10 #Batch the data points into 5000 and run on all cores for faster
    preprocessing
11 text = [str(doc) for doc in nlp.pipe(brief_cleaning1, batch_size
    =5000, n_threads=-1)]
12
13 #Takes 7-8 mins
14 print('Time to clean up everything: {} mins'.format(round((time
    () - t) / 60, 2)))
15

```

```
16 >>>Time to clean up everything: 7.68 mins
```

Code 4: The data shape after basic cleaning

Same process is also done for summary part. After cleaning the batch of texts, it will be time to decide the length of summary for the length of text. For this particular example, it has been chosen for summary to be max 15 words for a text shorter than 100 words. After preparing the ideal text group that will be chosen to be trained in the model, some tokens will be used for summary part to make encoder decoder model understand what part of data they are in so that they can process it accordingly.

```
1 #Check how much % of summary have 0-15 words
2 cnt=0
3 for i in pre['cleaned_summary']:
4     if(len(i.split())<=15):
5         cnt=cnt+1
6 print(cnt/len(pre['cleaned_summary']))
7
8 >>>0.9978234465335472
9
10 #Check how much % of text have 0-70 words
11 cnt=0
12 for i in pre['cleaned_text']:
13     if(len(i.split())<=100):
14         cnt=cnt+1
15 print(cnt/len(pre['cleaned_text']))
16
17 >>>0.9578389933440218
```

Code 5: Check for Length in the cleaned texts

The most important part, however, is to build the actual seq2seq model from the scratch. For starters, the data will be split to train and validation values. Then, some tokenizers will be used to optimize word count in order to adjust layers once the model is built in order to understand the text sequences better. Then three encoders and a decoder seen as the optimal size of the seq2seq system. For this Project, another LSTM encoder had been added for experiment purposes only to be found out that it heavily slowed the whole process.

```

1 # Encoder
2 encoder_inputs = Input(shape=(max_text_len,))
3
4 #embedding layer
5 enc_emb = Embedding(x_voc, embedding_dim, trainable=True)(
    encoder_inputs)
6
7 #encoder lstm 1
8 encoder_lstm1 = LSTM(latent_dim, return_sequences=True,
    return_state=True, dropout=0.4, recurrent_dropout=0.4)
9 encoder_output1, state_h1, state_c1 = encoder_lstm1(enc_emb)
10
11 #encoder lstm 2
12 encoder_lstm2 = LSTM(latent_dim, return_sequences=True,
    return_state=True, dropout=0.4, recurrent_dropout=0.4)
13 encoder_output2, state_h2, state_c2 = encoder_lstm2(
    encoder_output1)
14
15 #encoder lstm 3
16 encoder_lstm3=LSTM(latent_dim, return_state=True,
    return_sequences=True, dropout=0.4, recurrent_dropout=0.4)
17 encoder_outputs, state_h, state_c= encoder_lstm3(encoder_output2
    )
18
19 # Set up the decoder, using 'encoder_states' as initial state.
20 decoder_inputs = Input(shape=(None,))
21
22 #embedding layer
23 dec_emb_layer = Embedding(y_voc, embedding_dim, trainable=True)
24 dec_emb = dec_emb_layer(decoder_inputs)
25
26 decoder_lstm = LSTM(latent_dim, return_sequences=True,
    return_state=True, dropout=0.4, recurrent_dropout=0.2)
27 decoder_outputs, decoder_fwd_state, decoder_back_state =
    decoder_lstm(dec_emb, initial_state=[state_h, state_c])
28
29 #dense layer
30 decoder_dense = TimeDistributed(Dense(y_voc, activation='
    softmax'))
31 decoder_outputs = decoder_dense(decoder_outputs)
32
33 # Define the model
34 model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

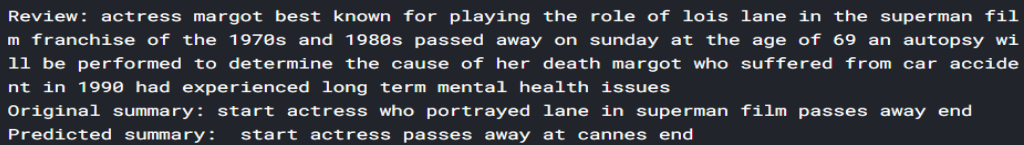
```

Code 6: Model architecture

Fitting the data to the model in a way that is most efficient proved to be the hardest part of the example. Even with a comparably high spec computer, fitting the data to an epoch with a batch size of 128 took around

two hours with a loss value of 6. Within this original experiment, the model used around 25 epochs to a success, which showed the capabilities and time given to that model. This really showed that the optimization needs of a model really will differ for the specs of the computer.

The last step will be building another encoder-decoder setup in order to predict the summary from the model's output vectors. By using the previous steps of the model with holding the information of the states of model, the decoder can create an output sequence that can be translated as the predicted value of summarization.



```
Review: actress margot best known for playing the role of lois lane in the superman film franchise of the 1970s and 1980s passed away on sunday at the age of 69 an autopsy will be performed to determine the cause of her death margot who suffered from car accident in 1990 had experienced long term mental health issues
Original summary: start actress who portrayed lane in superman film passes away end
Predicted summary: start actress passes away at cannes end
```

Figure 11: Result of a text summary using Seq2Seq

## 5.2 The Difficulties and Variables of Seq2Seq

As it is with everything involved coding, there were some difficulties involving performance issues, model design and understanding the general concept of what this example is all about. For the first experiments, it was quite difficult to understand how the process went along since the model is quite complicated, but a way had found to work it out. Performance wise, it wasn't great either. This will be tried to be improved with the next models that will be worked on.

It has been observed that size of model, number of layers, amount of encoder-decoders and the general weight of the model has quite the impact over the efficiency of a model. How positive or negative they are will be looked upon and will be checked upon within time.

## 5.3 Its Purpose In Text Summarization

With its generative nature, seq2seq model enables its users to build networks that are capable of creating a summary out of salvaging the already given text. This abstractive approach of seq2seq model takes text summarization to the next level compared to TextRank.

There are also different types of abstractive summarization other than using seq2seq model. But by being the most open source and flexible to use,

seq2seq methods can be integrated into different types of summarization techniques.

## 6 Named Entity Recognition(NER)

NER is an information extraction technique to identify and classify named entities in text. These entities can be pre-defined and generic like location names, organizations, time and etc, or they can be very specific like the example with the resume. By categorizing words, one would be fair to think that the summarization made with NLP intent would come across easier. Within the next stages of this Project, NER is a topic that is wanted to be delved upon to hopefully upgrade any potential text summarization product. This summary has been inspired from Nasir Safdari's NER work[Saf].

## 7 Transformers

Transformers is a transduction model, which, in NLP terms, takes a sequence input to create a sequence output. It provides Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch. It is like seq2seq in regards to its input output but uses pre built libraries to complete this job. Its main field of usages are autocorrect, translation and text summarization. Nowadays, transformers is mainly used within BERT, as a playing role within Google search engine and translation works. The ways transformers can be superior to the models built by RNN's and LSTM's are:

- They can be faster
- Parallel programming is more ideal
- Long term memory dependency is not that big of a problem compared to them.

Transformers is also made from an encoder and decoder. Like before, an encoder will take sequence input data, i.e. context vector, to change it to an output sequence. With an attention mechanism, the results that transformers model can get could be even more precise compared to other models. More extensive information can be found in Rohan Jagtap's Abstractive Text Summarization Using Transformers article [Jag].

```

1 # Transformers , pre-trained model
2
3 !pip install transformers
4
5 from transformers import pipeline
6
7 summarizer = pipeline("summarization")
8
9 ARTICLE = """ New York (CNN)When Liana Barrientos was 23 years
    old , she got married in Westchester County, New York.
10 A year later , she got married again in Westchester County, but
    to a different man and without divorcing her first husband.
11 Only 18 days after that marriage , she got hitched yet again.
    Then , Barrientos declared "I do" five more times , sometimes
    only within two weeks of each other.
12 In 2010 , she married once more , this time in the Bronx . In an
    application for a marriage license , she stated it was her "
    first and only" marriage .
13 Barrientos , now 39 , is facing two criminal counts of "offering a
    false instrument for filing in the first degree , " referring
    to her false statements on the
14 2010 marriage license application , according to court documents .
15 Prosecutors said the marriages were part of an immigration scam .
16 On Friday , she pleaded not guilty at State Supreme Court in the
    Bronx , according to her attorney , Christopher Wright , who
    declined to comment further .
17 After leaving court , Barrientos was arrested and charged with
    theft of service and criminal trespass for allegedly sneaking
    into the New York subway through an emergency exit , said
    Detective
18 Annette Markowski , a police spokeswoman . In total , Barrientos
    has been married 10 times , with nine of her marriages
    occurring between 1999 and 2002 .
19 All occurred either in Westchester County , Long Island , New
    Jersey or the Bronx . She is believed to still be married to
    four men , and at one time , she was married to eight men at
    once , prosecutors say .
20 Prosecutors said the immigration scam involved some of her
    husbands , who filed for permanent residence status shortly
    after the marriages .
21 Any divorces happened only after such filings were approved . It
    was unclear whether any of the men will be prosecuted .
22 The case was referred to the Bronx District Attorney\ 's Office
    by Immigration and Customs Enforcement and the Department of
    Homeland Security\ 's
23 Investigation Division . Seven of the men are from so-called "red
    -flagged" countries , including Egypt , Turkey , Georgia ,
    Pakistan and Mali .
24 Her eighth husband , Rashid Rajput , was deported in 2006 to his

```

```

    native Pakistan after an investigation by the Joint Terrorism
    Task Force.
25 If convicted, Barrientos faces up to four years in prison. Her
    next court appearance is scheduled for May 18.
26 """
27
28 print(summarizer(ARTICLE, max_length=130, min_length=30,
    do_sample=False))
29
30 >>> [{ 'summary_text': ' Liana Barrientos, 39, is charged with
    two counts of "offering a false instrument for filing in the
    first degree" In total, she has been married 10 times, with
    nine of her marriages occurring between 1999 and 2002 . At
    one time, she was married to eight men at once, prosecutors
    say .' }]

```

Code 7: Result from a transformers process

There are many complex applications and methods that use transformers that haven't been said here. After finding its capabilities and general power more Experiments regarding text summarization will be done in the future.

## 8 Discussion

Although the subject itself isn't unknown to the authors, it wouldn't be wrong to say that it has been challenging to gather information and find examples and datas that are helpful to project's cause. There have been problems in terms of availability of machine power when models needed to be worked but a way has been found so far and hopefully is going to be smooth as expected. There are still some algorithms that is wanted to be worked on, or at least experience if it is closed source(BERT, VizSeq...) but first it is needed to understand completely what is in the hand.

Coding-wise, there are still some aspects that have been not familiar enough to authors to feel comfortable. It needs time and many errors along the way to find the right path for the clear picture of blueprint of the final model.

### 8.1 Current Results That Have Concluded

For the project, only TextRank and Seq2Seq have been looked extensively. For the moment, The conclusion on them both is that Seq2Seq is, while less flexible in terms of data and model weight, is more fitting when thought about text summarization compared to TextRank. There is also a downside of TextRank when size of a singular text comes to mind, which if it isn't

longer and doesn't have many words. It basically becomes incompetent to tell which words are important. Same sort of problem can be occurred in Seq2Seq but rather than a single text, if all given texts are not a big structure, the loss value becomes quite big, thus making it useless in summary department. Something that also needs to be cautious of is to find the right model. If there is even a tiny bit unnecessary information given to or built to model, the whole process becomes much, much slower. Since its the early legs of the project, there are still more experiments to do to get the optimal settings. Once these experiments are done, it will be easier to decide which algorithm is superior.

## 8.2 Risk Analysis

There are some difficulties that may happen that is already discussed in the previous parts of the discussion, but the major one is to managing time and resources when a network model process begins. This is why whenever a model is built, if it looks or feels like it is underperforming, then it must be assumed as a failure before repeating the same mistake and try to better the model as a whole with different settings. Resource-wise, not every computer will be able to perform deep learning tasks. That is why if the power of the personal computers seems not enough, resources like google colab and virtual machines will be used.

There is also the difficulty of finding data for models without being repetitive or using other people's creation. The solution to that is to regularly salvage data from different websites. There is also the matter of how the product will work. The planned way, using FastAPI, will most likely work. But if there is a problem, the authors will try to find a way to solve it. Finally, the question that whether the model will be able to perform within Turkish language. This, even with the research, is still an unknown. But hopefully, within the next steps of this project, it will be accomplished

## 9 Conclusion

The aim is to create a product so by looking at the already made products, what works and what doesn't when it comes to summarization. By collecting the data about the strengths and weaknesses of models, it will be easier to create a model that will work in optimum fashion. It will also help to integrate the model to turkish language, which is the important part of the aimed product.



## References

### References

- [Jag] Rohan Jagtap. *Abstractive Text Summarization Using Transformers*. URL: <https://medium.com/swlh/abstractive-text-summarization-using-transformers-3e774cc42453>. (Accessed 1 January 2021).
- [Mis] Madhav Mishra. *Seq2seq: Abstractive Summarization Using LSTM And Attention Mechanism [CODE]*. URL: <https://medium.com/analytics-vidhya/seq2seq-abstractive-summarization-using-lstm-and-attention-mechanism-code-da2e9c439711>. (Accessed 1 January 2021).
- [Ola] Christopher Olah. *Understanding LSTM Networks*. URL: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. (accessed: 01.01.2021).
- [Saf] Nasir Safdari. *Named Entity Recognition (NER) with keras and tensorflow*. URL: <https://towardsdatascience.com/named-entity-recognition-ner-meeting-industrys-requirement-by-applying-state-of-the-art-deep-698d2b3b4ede>. (Accessed 1 January 2021).

Original work for Seq2Seq Model:

<https://www.kaggle.com/sandeepbhogaraju/text-summarization-with-seq2seq-model>