İstanbul
Bilgi Üniversitesi

Title: Comparison of the Text Summarization Algorithms

by

Tuğra Burak Çakıcı, 117200073
Arda Cem Özmen, 115200046

Supervised by

Uzay Çetin

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

Jun, 2021

Abstract

Text summarization, as far as Natural Language Processing community and their studies are concerned, is a complex task to deal with. Although the basic logic of summarization is simple, there are multiple different variables which would affect the process dearly. These variables can make summarization better,worse or simply halt the whole process if not looked after carefully. There are also different types of methods that one can use when tasked with this job that offer different approaches with different results. The main aim of this project is to differentiate these methods both singularly and in the bracket of the algorithms that do the same job. Methods that are chosen for this project, TextRank, Seq2Seq and Transformer, are deemed to be the most popular ones among the NLP community. By training and creating optimum environments for these algorithms, the previously mentioned variables will be clearer to adjust for the specific methods. Once these tasks are over, these optimum models are to be deployed for creating an overall product with the help of FastAPI. At the end of this study, one will be able to compare these algorithms by their strengths and weaknesses.

Table of Contents

List of Figures

1 Introduction

It is fair to say that the advancements regarding AI and AI like softwares are in an all-time high nowadays. Multifunctional robots, language translation systems independent of outside vocabulary, face recognition patterns and many more examples are developed more frequently compared to previous years. The main reason behind this steady yet big jump in the evolution of these akin products is that there are even more methods and tools to get the desired result using newer technologies with better output and lesser resource and time. So whenever one tries to find the best solution for their product or problem, it becomes a pick-of-the-bunch situation that a relatively newcomer or average user would have no idea which one to choose.

The main idea of this project is also related to the matter that is talked about earlier. The objective is aimed to be one of the harder tasks of NLP and ML/DL in general, summarization of texts. Summarization is particularly difficult because generation of texts without a precise direction will end up in a meaningless sentences in most times. Since, as talked about before, there are many different ways to find a summarization algorithm that works within desired output despite the difference in techniques. Although the results and accuracies may wary with different algorithms, the core idea will be the same and the summarization of the text within some kind will be the objective.

The plan is to test out three popular choices of algorithms for this job that will be talked about, implement them for both Turkish and English languages and find the main differences of the methods that have been used with different criterias, ultimately making a comparison out of them. All of this will culminate with models being deployed for usage for people to see which one they like the most within their own judgement.

2 Background

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

There are two main parts of the coding progress. The first one involves research about models and their implementation. How would they react to experimental changes and how to deploy them when all is said and done.

The second part involves preparing the data to feed it to models. It is imperial that models are fed as good of a data as possible. From labeling them to cleaning, this part is maybe the most crucial part of the whole project

2.3 Data Preparation

For all neural network and alike models, data is absolutely the focal point of the system. It defines the capabilities and end results for the given entry points. Since two of the models that will be talked about require excessive amounts of data, it was only natural that data needed to be gathered. All the datasets will be cleaned using regex.

There are 3 main datasets that were found and gathered for the processing. First one is the News Summary data that includes main text that includes full story and its summarized version. The data is divided by the original creator by, seemingly, their size, meaning one includes longer news(most of them over 200 words) while other one includes moderate sized news(around 50-100 words). The Seq2Seq model that have been implemented used the moderate sized version while longer version is used by t5 model.

Second dataset is CNN news dataset that is semi-publicly published and used by many NLP users in their assets. Original data was as long as 900.000 pairs of news and its summary but the most that could have been pulled for this project was around 100.000.

The third one is a dataset that is scrapped by the authors of this project for Turkish language purposes. It includes an archive of news and their headlines, with headlines acting as the summary of the text. The dataset has over 130000 unique news branching from 2020 to 2017, with an average mean of 45-60 words among them. This is the only data that cannot be referenced within the paper because of the legal rules of scrapping and its non-profit and confidentiality agreements. The other two will be referenced at the end. Scraper of this data will be shared only for the project related reasons.

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 recurrent 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 recurrently give information within its hidden states from the previous models. With their recurrent system, text summarization objective becomes much easier.

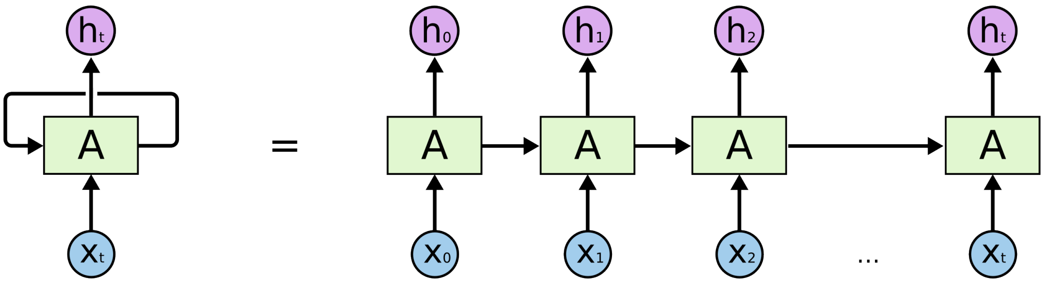


Figure 1: An unrolled RNN

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.

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.

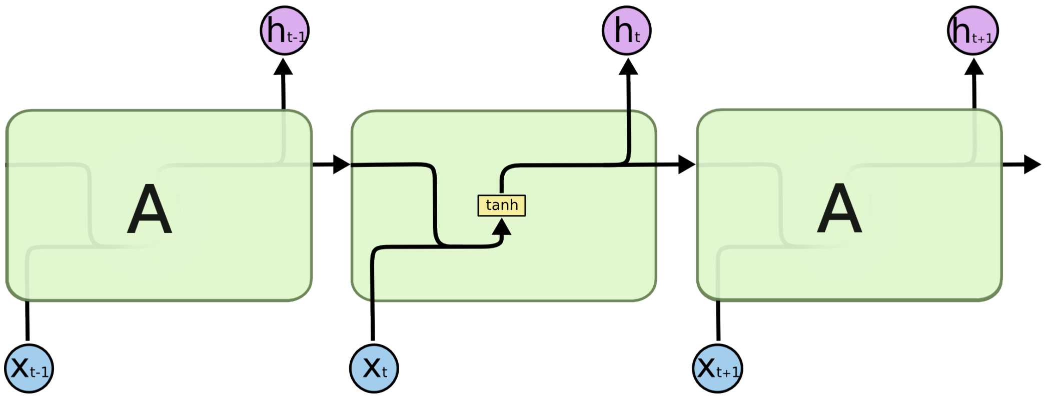


Figure 2: A Simple RNN

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.

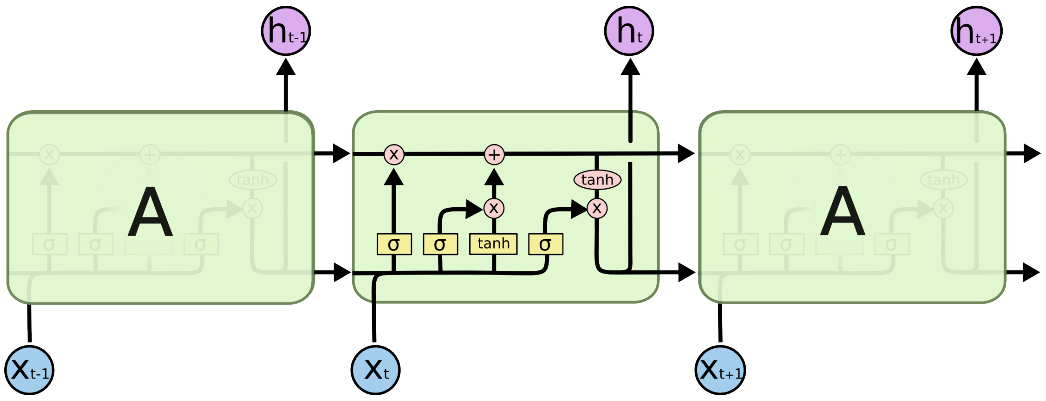


Figure 3: LSTM module

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.

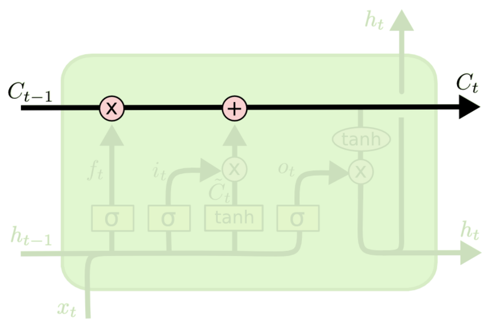
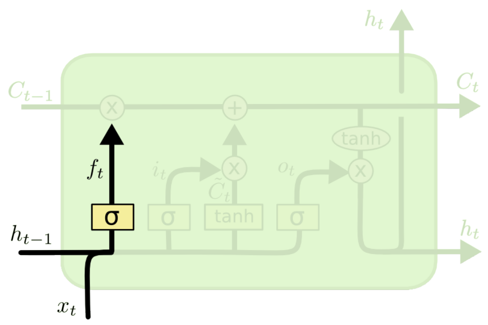


Figure 4: LSTM cell state

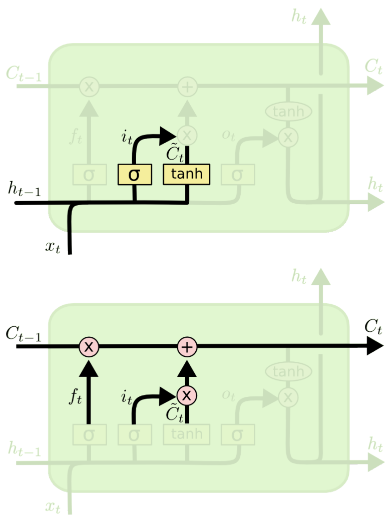
Next part is the forget gate layer, where a sigmoid decides whether to keep the information coming from the previous state or not. It will either give 1, i.e. keep, or 0, i.e. forget, and carry the decision to the cell state. This information can be about a label value that may become important in future.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



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

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.

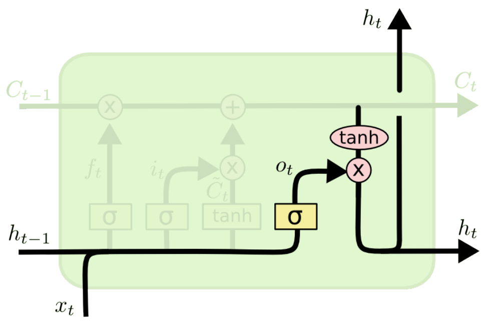


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 6: LSTM input gate

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.



$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 7: LSTM output layer

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 Summarizaton 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.

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, Transformer 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. The similarity scores are calculated using PageRank algorithm that have been produced for the website references.

4.1 PageRank

PageRank is an algorithm that has been used to sub-reference other websites based on the content within the processed page. Meaning, it looks for cited websites within the processed page to find the page that has been cited the most out of them. The score of a PageRank value is considered as the probability that the user will visit that website, with the main idea being 'important pages are connected to important pages'. One of the most important places that this algorithm has been used is Google Search. Of course, Google's algorithm is considered to be way more complicated compared to the normal ones.

The equation behind PageRank should also be talked about. Let's assume there are websites A, B, C, D and consider them as seeds. Page A has websites (T1...TN) that point towards it. There is also a dampening factor d that is used to calibrate the probability factor. It usually is considered as 0.85. The number of pages that come from page A, C(A), should also be considered before the equation. All in all, the equation for page A PageRank score will be calculated like this

$$PRA = (1 - d) + d(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

This equation will result a distribution value using the capitalized dampening value for page A. This value will be 1 for all websites. Meaning, page B's PageRank value is the result of the multiplication of mentioned dampening factor using page A. If B connects to C after A, the value of PageRank will be multiplied by the dampening factor again(0.85*0.85=0.7225).

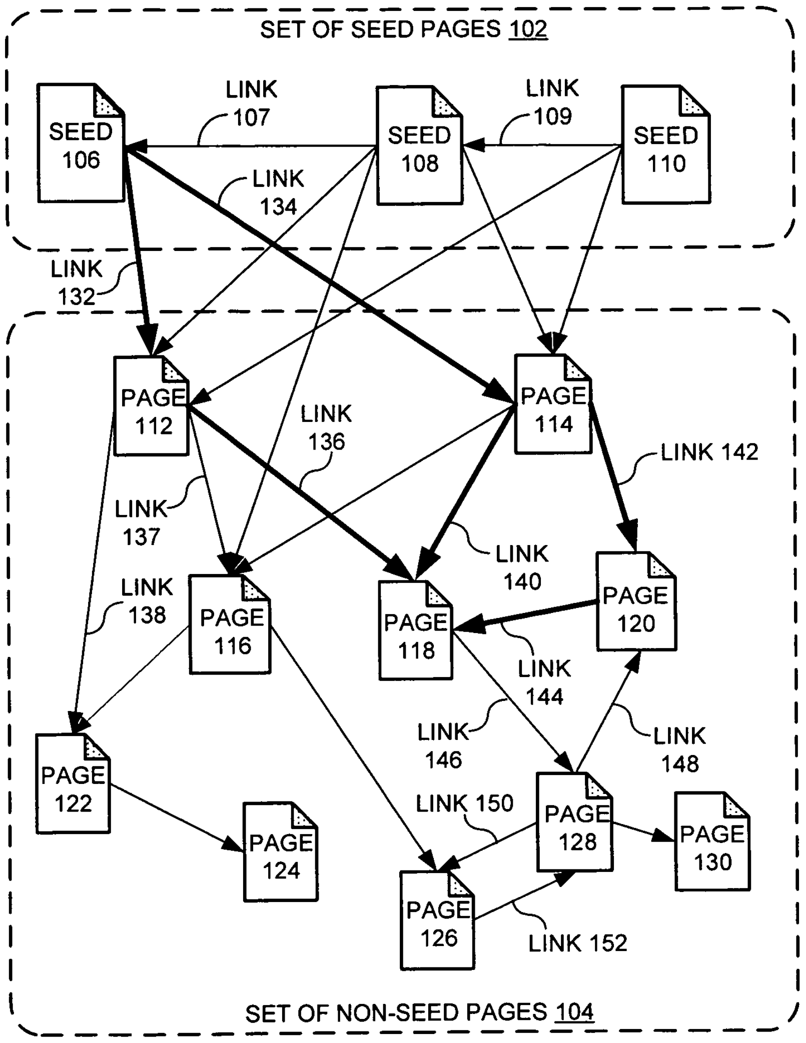


Figure 8: PageRank Algorithm[Haj]

4.2 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, like in PageRank with websites, a chart is created with seeds, with each sentence is a seed, these seeds are linked by weight and similarity. A simple outlook between sentences A and B, where the similarities are being looked as their connection in PageRank, should look like this:

$$(0 <= similarity(A, B) <= 1)$$

$$(similarity(A, A) = 1)$$

$$(similarity(A, B) = / = 1) => (A = / = B)$$

Similarity is looked for the words and cosine similarity is used to calculate the references of the words. If the word vectors are the same, the score will be 1 otherwise 0. For all the words within the sentence, these values will get added into similarity matrix and the highest values out of all of them will be considered as the projected and wanted output.

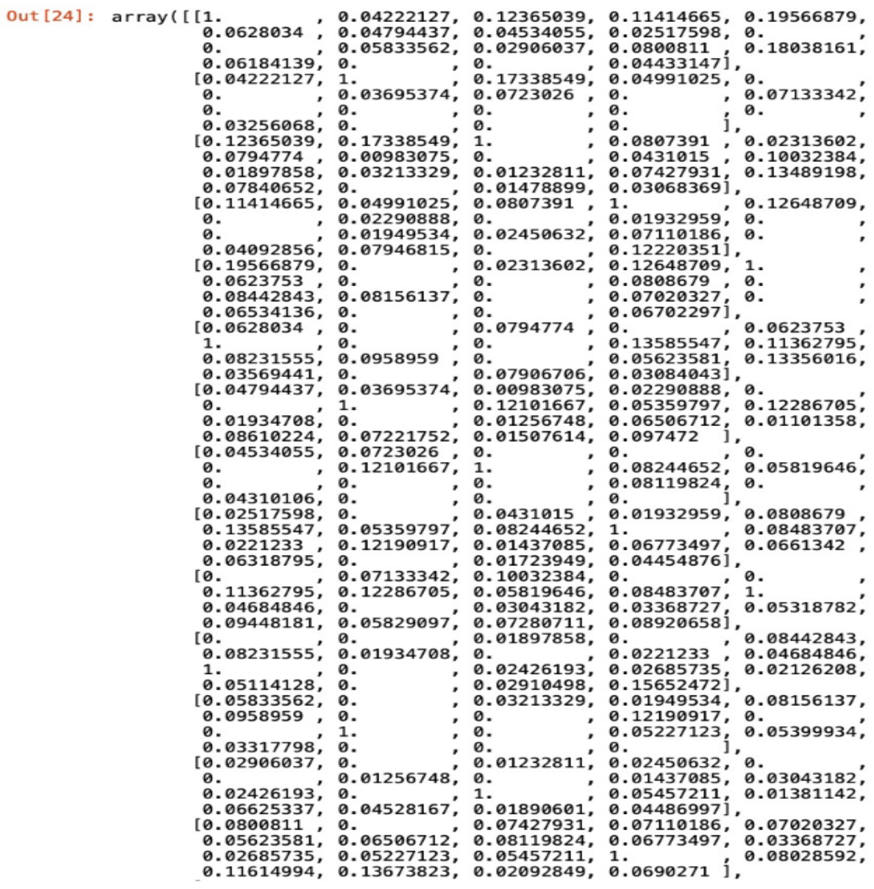
4.3 An Example of TextRank Text Summarization

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.

```
from collections import Counter
def bag_of_words(sentence):
    return Counter(word.lower().strip('.,') for word in sentence
   .split(' '))
    #This part will count the words given in a sentence. It will
     be useful for PageRank part when the importance of words
   gets evaluated.
bag_of_words(sentences[0])
>>> Counter({'the': 2,
          'covid-19': 1,
          'pandemic': 3,
          'also': 1,
          'known': 1,
          'as': 1,
          'coronavirus': 3,
          'is': 1,
          'an': 1,
          'ongoing': 1,
          'of': 1,
          'disease': 1,
          '2019': 1,
          '(covid-19)': 1,
          'caused': 1,
          'by': 1,
          'severe': 1,
          'acute': 1,
          'respiratory': 1,
          '2': 1,
          '(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.        , 0.04222127, 0.12365039, 0.11414665, 0.19566879,
                 0.0628034 , 0.04794437, 0.04534055, 0.02517598, 0.        ,
                 0.        , 0.05833562, 0.02906037, 0.0800811 , 0.18038161,
                 0.06184139, 0.        , 0.        , 0.04433147],
                [0.04222127, 1.        , 0.17338549, 0.04991025, 0.        ,
                 0.        , 0.03695374, 0.0723026 , 0.        , 0.07133342,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.03256068, 0.        , 0.        , 0.        ],
                [0.12365039, 0.17338549, 1.        , 0.0807391 , 0.02313602,
                 0.0794774 , 0.00983075, 0.        , 0.0431015 , 0.10032384,
                 0.01897858, 0.03213329, 0.01232811, 0.07427931, 0.13489198,
                 0.07840652, 0.        , 0.01478899, 0.03068369],
                [0.11414665, 0.04991025, 0.0807391 , 1.        , 0.12648709,
                 0.        , 0.02290888, 0.        , 0.01932959, 0.        ,
                 0.        , 0.01949534, 0.02450632, 0.07110186, 0.        ,
                 0.04092856, 0.07946815, 0.        , 0.12220351],
                [0.19566879, 0.        , 0.02313602, 0.12648709, 1.        ,
                 0.0623753 , 0.        , 0.        , 0.0808679 , 0.        ,
                 0.08442843, 0.08156137, 0.        , 0.07020327, 0.        ,
                 0.06534136, 0.        , 0.        , 0.06702297],
                [0.0628034 , 0.        , 0.0794774 , 0.        , 0.0623753 ,
                 1.        , 0.        , 0.        , 0.13585547, 0.11362795,
                 0.08231555, 0.0958959 , 0.        , 0.05623581, 0.13356016,
                 0.03569441, 0.        , 0.07906706, 0.03084043],
                [0.04794437, 0.03695374, 0.00983075, 0.02290888, 0.        ,
                 0.        , 1.        , 0.12101667, 0.05359797, 0.12286705,
                 0.01934708, 0.        , 0.01256748, 0.06506712, 0.01101358,
                 0.08610224, 0.07221752, 0.01507614, 0.097472  ],
                [0.04534055, 0.0723026 , 0.        , 0.        , 0.        ,
                 0.        , 0.12101667, 1.        , 0.08244652, 0.05819646,
                 0.        , 0.        , 0.        , 0.08119824, 0.        ,
                 0.04310106, 0.        , 0.        , 0.        ],
                [0.02517598, 0.        , 0.0431015 , 0.01932959, 0.0808679 ,
                 0.13585547, 0.05359797, 0.08244652, 1.        , 0.08483707,
                 0.0221233 , 0.12190917, 0.01437085, 0.06773497, 0.0661342 ,
                 0.06318795, 0.        , 0.01723949, 0.04454876],
                [0.        , 0.07133342, 0.10032384, 0.        , 0.        ,
                 0.11362795, 0.12286705, 0.05819646, 0.08483707, 1.        ,
                 0.04684846, 0.        , 0.03043182, 0.03368727, 0.05318782,
                 0.09448181, 0.05829097, 0.07280711, 0.08920658],
                [0.        , 0.        , 0.01897858, 0.        , 0.08442843,
                 0.08231555, 0.01934708, 0.        , 0.0221233 , 0.04684846,
                 1.        , 0.        , 0.02426193, 0.02685735, 0.02126208,
                 0.05114128, 0.        , 0.02910498, 0.15652472],
                [0.05833562, 0.        , 0.03213329, 0.01949534, 0.08156137,
                 0.0958959 , 0.        , 0.        , 0.12190917, 0.        ,
                 0.        , 1.        , 0.        , 0.05227123, 0.05399934,
                 0.03317798, 0.        , 0.        , 0.        ],
                [0.02906037, 0.        , 0.01232811, 0.02450632, 0.        ,
                 0.        , 0.01256748, 0.        , 0.01437085, 0.03043182,
                 0.02426193, 0.        , 1.        , 0.05457211, 0.01381142,
                 0.06625337, 0.04528167, 0.01890601, 0.04486997],
                [0.0800811 , 0.        , 0.07427931, 0.07110186, 0.07020327,
                 0.05623581, 0.06506712, 0.08119824, 0.06773497, 0.03368727,
                 0.02685735, 0.05227123, 0.05457211, 1.        , 0.08028592,
                 0.11614994, 0.13673823, 0.02092849, 0.0690271 ],
```

Figure 9: 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.

```
import networkx as nx
nx_graph = nx.from_scipy_sparse_matrix(similarity_graph)
scores = nx.pagerank(nx_graph)
scores

>>> {0: 0.058607608354242294,
 1: 0.04601850507134548,
 2: 0.057010612605271425,
 3: 0.05146310022030544,
 4: 0.05292206864494559,
 5: 0.055478147303211045,
 6: 0.05232471287928315,
 7: 0.04650125802396069,
 8: 0.05514990276837194,
 9: 0.05748267984597134,
 10: 0.04760188651193677,
 11: 0.046632574078683535,
 12: 0.04481193684950849,
 13: 0.060230270320510274,
 14: 0.056320045974837064,
 15: 0.061130538190084016,
 16: 0.04689382572969491,
 17: 0.04587183540789658,
 18: 0.0575484912199396}
```

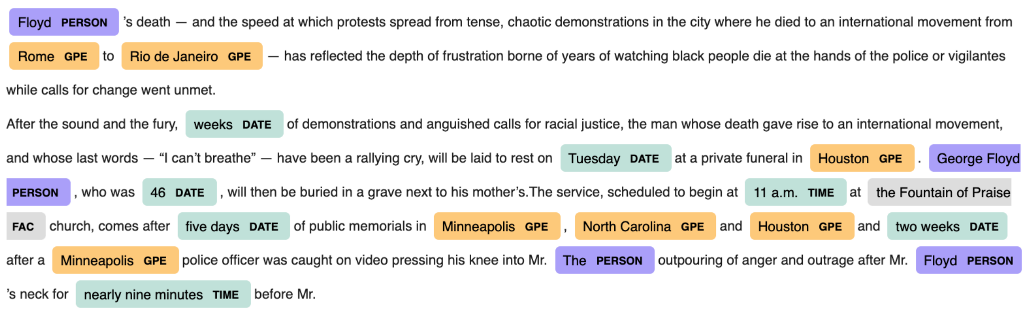
Code 2: PageRank results

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

```
print(textrank(document))
>>> [(0.061130538190084016,
  'It has led to the postponement or cancellation of events,
    widespread supply shortages exacerbated by panic buying,
    agricultural disruption and food shortages, and decreased
    emissions of pollutants and greenhouse gases.'),
 (0.060230270320510274,
  'Many places have also worked to increase testing capacity and
     trace contacts of the infected.'),
 (0.058607608354242294,
  'The COVID-19 pandemic, also known as the coronavirus pandemic
    , is an ongoing pandemic of coronavirus disease 2019 (COVID
    -19) caused by severe acute respiratory syndrome coronavirus
    2 (SARS-CoV-2).'),
 (0.0575484912199396,
  'There have been incidents of xenophobia and discrimination
    against Chinese people and against those perceived as being
    Chinese or as being from areas with high infection rates.'),
 (0.05748267984597134,
  "Recommended preventive measures include social distancing,
    wearing face masks in public, ventilation and air-filtering,
    hand washing, covering one's mouth when sneezing or coughing,
     disinfecting surfaces, and monitoring and self-isolation for
     people exposed or symptomatic."),
 (0.057010612605271425,
  'The World Health Organization declared the outbreak a Public
    Health Emergency of International Concern in January 2020 and
     a pandemic in March 2020.'),
 (0.056320045974837064,
  'The pandemic has caused global social and economic disruption
    , including the largest global recession since the Great
    Depression.'),
 (0.055478147303211045,
  'The virus spreads mainly through the air when people are near
    each other.'),
 (0.05514990276837194,
  'People remain infectious for up to two weeks, and can spread
   the virus even if they do not show symptoms.'),
 (0.05292206864494559,
  'Symptoms of COVID-19 are highly variable, ranging from none
   to severe illness.'),
 (0.052324712879283315,
  '[b It leaves an infected person as they breathe, cough,
   sneeze, or speak and enters another person via their mouth,
   nose, or eyes.'),
 (0.05146310022030544,
  'As of 28 December 2020, more than 81.1 million cases have
   been confirmed, with more than 1.77 million deaths attributed
    to COVID-19.'),
 (0.04760188651193677,
  'Several vaccines are being developed and distributed.'),
 (0.04689382572969491,
  'Many educational institutions have been partially or fully
   closed.'),
 (0.046632574078683535,
  'Current treatments focus on addressing symptoms while work is
    underway to develop therapeutic drugs that inhibit the virus
   .'),
]
```
Code 3: TextRank Output

4.4 NER Implementation

Named Entity Recognition, NER for short, is a classification method that looks for the entities within the given texts. Entities will either be defined by the default or the user will define them themselves. TextRank, in theory, is the model that could benefit the most from a technology like this. After partitioning the data into a singular subject and training labels for that particular subject, with giving more weights into these values, TextRank's overall score would surely increase.



Floyd PERSON 's death — and the speed at which protests spread from tense, chaotic demonstrations in the city where he died to an international movement from Rome GPE to Rio de Janeiro GPE — has reflected the depth of frustration borne of years of watching black people die at the hands of the police or vigilantes while calls for change went unmet.

After the sound and the fury, weeks DATE of demonstrations and anguished calls for racial justice, the man whose death gave rise to an international movement, and whose last words — “I can’t breathe” — have been a rallying cry, will be laid to rest on Tuesday DATE at a private funeral in Houston GPE . George Floyd PERSON , who was 46 DATE , will then be buried in a grave next to his mother’s.The service, scheduled to begin at 11 a.m. TIME at the Fountain of Praise FAC church, comes after five days DATE of public memorials in Minneapolis GPE , North Carolina GPE and Houston GPE and two weeks DATE after a Minneapolis GPE police officer was caught on video pressing his knee into Mr. The PERSON outpouring of anger and outrage after Mr. Floyd PERSON 's neck for nearly nine minutes TIME before Mr.

Figure 10: NER on a Result of TextRank

5 Sequence2Sequence

Sequence To Sequence(Seq2Seq for short) is an LSTM based encoder decoder architecture that can be used with many learning method and for many ML/DL related objectives. As it is in the name, Seq2Seq models take an input sequence that is accordingly matchable with the data that it has been trained and creates an output sequence that has learned from the previous results that given by the data. For the output that is desired to be learnt by step by step and remember the important portions of the given information, Seq2Seq can be seen as a great method to be involved in. Although this paper will cover up an experiment that is more related to NLP, Seq2Seq models can be used in variety of different jobs and processes, such as language translation, image captioning, conversational models and many more.

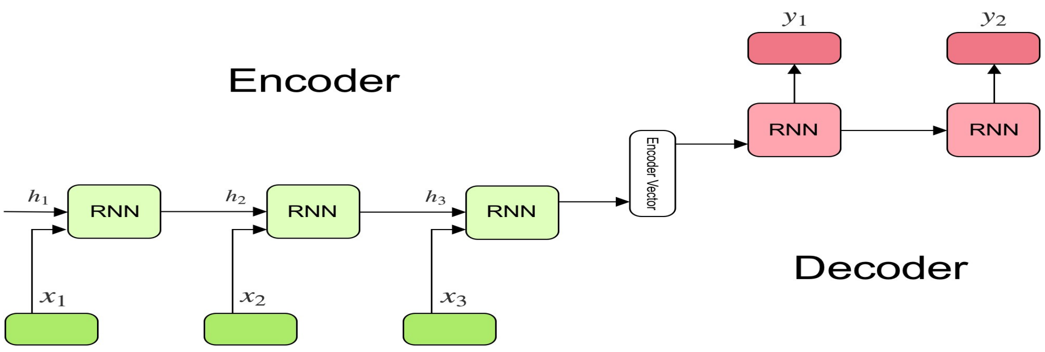


Figure 11: A basic Seq2Seq model

5.1 The Process of a Seq2Seq System

The starting step of the whole process is to transform input and output values to numeric sequential values. Within this Project, the sentences will be turned into numeric outputs that represent each word that sentence contains thanks to the respective tokenizers assigned to data. If X and Y are assumed as the input output sentences, then x=[x1,x2,x3..xn] and y=[y1,y2,y3..yn] are to be considered as the sequential value equals of the original data. 'n' value will be determined as the max length that will be given to targeted value, for these experiments it will be word count, and output value sequences will include a start-end like substringing, in this case it is sostok-eostok, that will make output sequence to be more recognizable to the model.

The general idea behind the prediction system is to find the probabilities of softmax applied input output results that makes the most sense compared to the true ones. The generated output y value that is closest to the actual Y value becomes acknowledged as the best prediction value. This value will be predicted with the help of LSTM's sequential nature, internal states, that enables it to store values in longer sequences to find out whether the information is needed or not. The sentence guessing will work word to word, meaning in every possible word, a probability comparison between words that seem most probable to be included . The most probable ones will create a sentence that is assumed by the correct answer by the model.

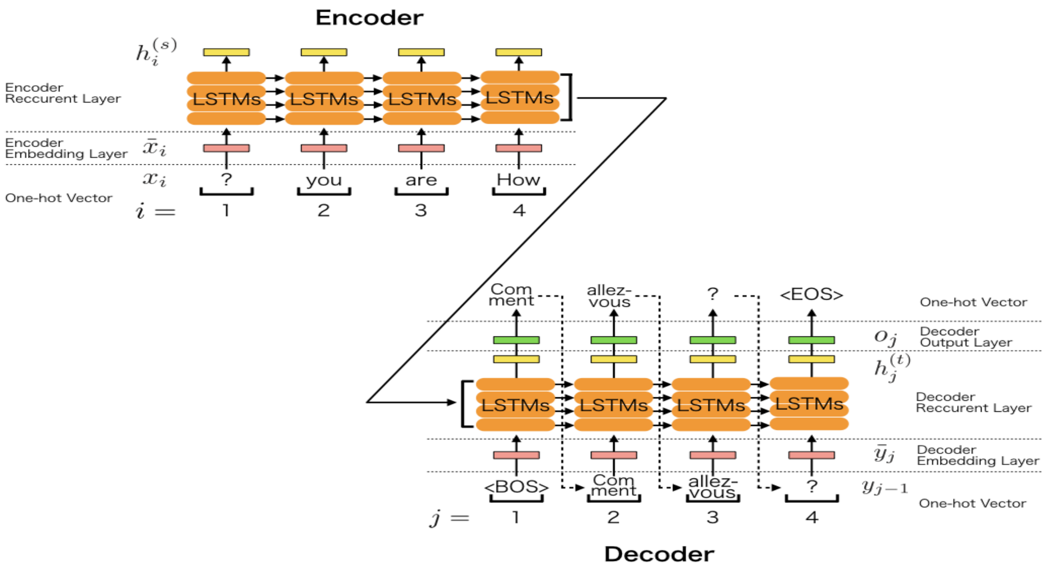


Figure 12: Detailed Seq2Seq for Translation

The architecture of these encoder decoder models is the most important thing in building a Seq2Seq model. Step by step, it all starts with an embedding layer that is defined by the desired encoder inputs. Embedding layer will turn sentences into embedding vectors, which will be taken by the LSTM layers that will process these vectors to create hidden vectors. These values will be important to remember later on when decoding sequence happens. Same procedure happens in decoder model with output values taking the spotlight.

Once the training of the model is over, encoder and decoder models will be available to process the input sequences to output sequences, creating a result. By using the states of both encoder and decoder, a function will be able to put together a string of predictions that can be considered as logical enough by the model. Of course, the answer is as logical as the model's capabilities trained by the data it fed on, but in its own terms, it is the best possible solution.

5.2 Attention Layer

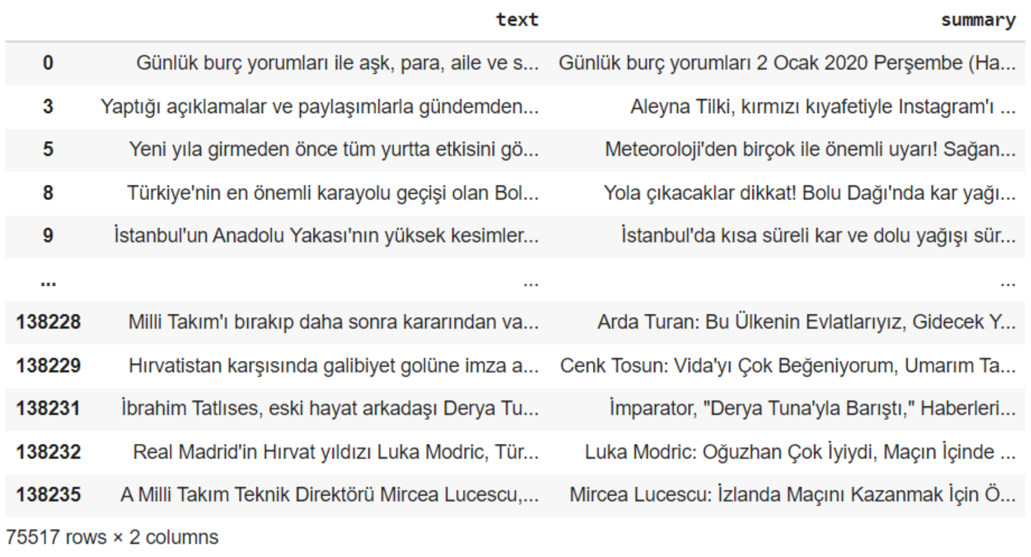
As it was talked about, the model will choose a sequence of words that it feels that makes the most sense via probability of words that it has the closest to the actual value that it was fed on via tokenizers. However, there may be times that model strays from the previous knowledge and forgets which parts that it needs to be focused on. This results in predictions that are very unrelated to the actual part of the aimed value. The solution for this problem in transformer models came in the way of masking, which hides the next value to be predicted within the word sequence. In Seq2Seq models, however, attention layer has been brought up as a legitimate solution for a problem that was in dire need of solving. This method will improve the output vector in a sense that it will be more alingable with the original aimed output.

The logic behind is that probability of that possible trajected word reflects the importance of that word vector respect previous hidden state, which holds information about the important parts that needs to be focused in order to predict next word within text vector. With this layer, decoder will be able to decide what part it needs to be focused, thanks to attention mechanism. This approach helps decoder to see and decide which words that will be used in a sequence of annotations, rather than trying to predict all the sentence at once.

This project will be implementing the attention layer idea that has been created by Dzmitry Bahdanau which specifically instructed to focus on tasks that NLP delve on. By deciding the next sequence from the energy(probability) of the previous pool of states, the implemented layer will help the decoder model to try to generate a sequence based on the word with the highest amount of energy. By the end of the sequence, the result will at least be somewhat logically comparable to the original one.

5.3 An Example of Seq2Seq Text Summarization

As it is in all learning algorithms, needed data is fed to the system to be processed. After many experiments and their, most of the time unfortunate, outcomes, the data is partitioned based on both its original text and its overall summary. Since the data uses headlines as its summary section, it has been decided that all summaries needed to be longer than 7 words. The number 7 is chosen to partition the data into something meaningful, as in anything more than it results in a significantly smaller data. The text length for the data partitioning has been chosen as minimum 22 words. This really has been chosen from the eye's perspective, believing that this is a minimum right amount to summarize. After a basic cleaning process, which the only reason it was basic is because tokenizers do not care about writing mishandlings and such and tokenize them as they are, the data will be deemed ready to be split into train and validation. If it is not clear from the picture, the data that has been used is scraped by the authors and is Turkish.



	text	summary
0	Günlük burç yorumları ile aşk, para, aile ve s...	Günlük burç yorumları 2 Ocak 2020 Perşembe (Ha...
3	Yaptığı açıklamalar ve paylaşımlarla gündemden...	Aleyna Tilki, kırmızı kıyafetiyle Instagram'ı ...
5	Yeni yıla girmeden önce tüm yurtta etkisini gö...	Meteoroloji'den birçok ile önemli uyarı! Sağan...
8	Türkiye'nin en önemli karayolu geçişi olan Bol...	Yola çıkacaklar dikkat! Bolu Dağı'nda kar yağı...
9	İstanbul'un Anadolu Yakası'nın yüksek kesimler...	İstanbul'da kısa süreli kar ve dolu yağışı sür...
...	...	...
138228	Milli Takım'ı bırakıp daha sonra kararından va...	Arda Turan: Bu Ülkenin Evlatlarıyız, Gidecek Y...
138229	Hırvatistan karşısında galibiyet golüne imza a...	Cenk Tosun: Vida'yı Çok Beğeniyorum, Umarım Ta...
138231	İbrahim Tatlıses, eski hayat arkadaşı Derya Tu...	İmparator, "Derya Tuna'yla Barıştı," Haberleri...
138232	Real Madrid'in Hırvat yıldızı Luka Modric, Tür...	Luka Modric: Oğuzhan Çok İyiydi, Maçın İçinde ...
138235	A Milli Takım Teknik Direktörü Mircea Lucescu,...	Mircea Lucescu: İzlanda Maçını Kazanmak İçin Ö...

75517 rows × 2 columns

Figure 13: Data header

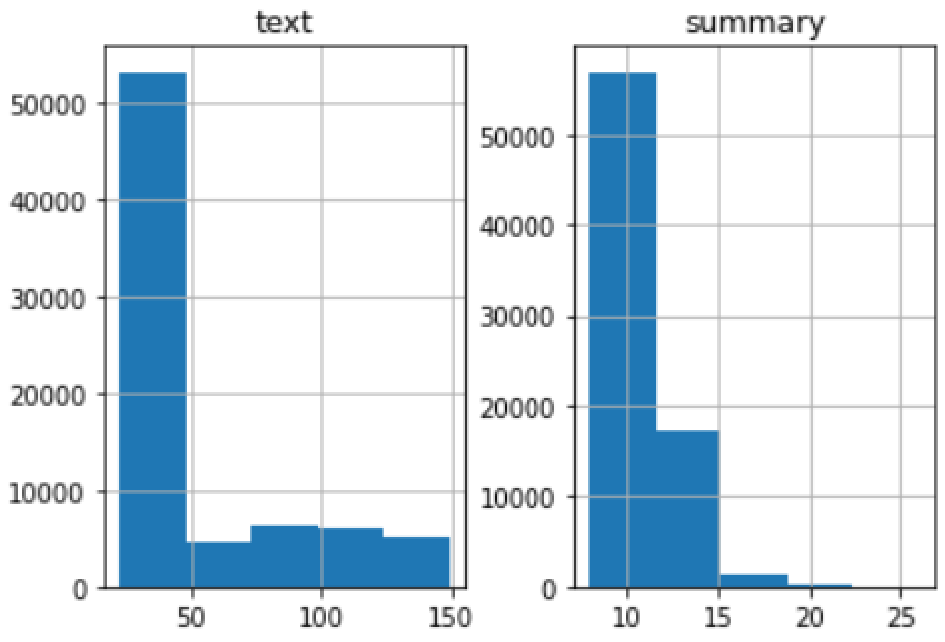
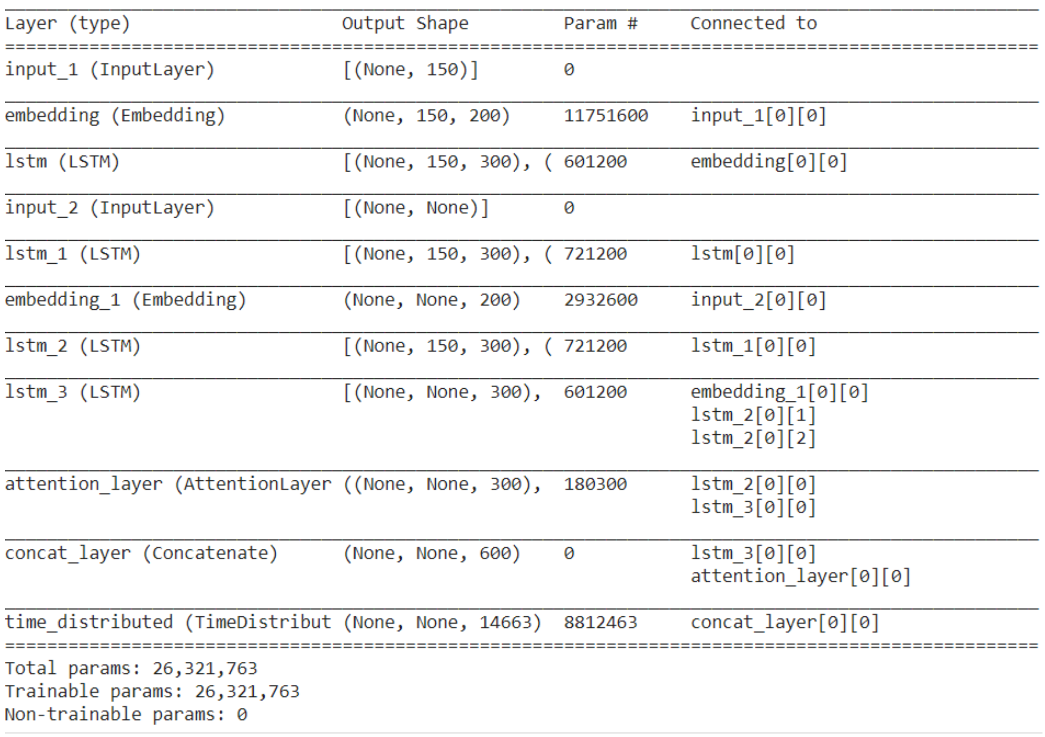


Figure 14: Data Length Representation

Before splitting the data into train and validation, a start and an end indicator, in this case sostok-eostok, is put for the decoder to understand where it starts and where it ends. This part will be explained at the latter stages. After indicating the data, it is split into train and test parts. These will go through a quick rundown of tokenizers, where words will be tokenized into numbers, put together as a sequence and have a fixed size defined by the padding.

After the job with data and their tokenizers are done, the model building for layers of initial encoder and decoder begins. It will consist of input layers that will be used in encoder decoder models, two embedding layers, which are there for model to learn the word embeddings that is provided by the data, for both encoder and decoder, lstm layers, which will be used to process the data within the model with its states to learn about the patterns, Bahdanau attention, which has been talked about already, Concatenate layer ,which will be used to combine the decoder and attention values, and a dense layer to softmax both decoder and attention results to create probability for the sequences. This will continue with the model being trained using sparse categorical crossentropy to observe loss values of summaries.



```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 150)]        0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 150, 200)     11751600    input_1[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     [(None, 150, 300), ( 601200      embedding[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 150, 300), ( 721200      lstm[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 200)    2932600     input_2[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 150, 300), ( 721200      lstm_1[0][0]
__________________________________________________________________________________________________
lstm_3 (LSTM)                   [(None, None, 300),  601200      embedding_1[0][0]
                                                                 lstm_2[0][1]
                                                                 lstm_2[0][2]
__________________________________________________________________________________________________
attention_layer (AttentionLayer ((None, None, 300),  180300      lstm_2[0][0]
                                                                 lstm_3[0][0]
__________________________________________________________________________________________________
concat_layer (Concatenate)      (None, None, 600)    0           lstm_3[0][0]
                                                                 attention_layer[0][0]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 14663)  8812463     concat_layer[0][0]
==================================================================================================
Total params: 26,321,763
Trainable params: 26,321,763
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 15: Model Summary

After the training is over, the trained states will be put under two new encoder and decoder models. The states that have been trained for the purpose to be included inside these new models will be perfect for validating the data. Their creation comes before defining two important functions that are the most important pieces of the puzzle. The first one is seq2summary, which will convert the validation data from its numpy array stage to an actual text. This has been managed by converting the sequence of tokens to their former state by the designated tokenizer. The second one, the most important one, decode sequence will try to guess a different vector sequence to the input one using the trained decoder. The figure under will show some results that have been solved by these two functions.

```
x=["Koronavirus teshisi konulduktan sonra hastaneye kald r lan
     ve solunum guclugu cektigi icin entube edilen Mehmet Ali
    Erbil, tedavisinin ardindan taburcu edildi. Cikista
    gazetecilerin sorularini yanitlayan Mehmet Ali, Ben Ingiltere
    'den gelen virusu gecirdim, cok agirdi. Gecirmeyenler ciddiye
     alsin dedi"]
print(x[0])
print("Predicted summary:",decode_sequence(tokenize(x).reshape
    (1,150)))
print(rouge.get_scores(decode_sequence(tokenize(x).reshape
    (1,150)), x[0]))

>>>Koronavirus teshisi konulduktan sonra hastaneye kald r lan
    ve solunum guclugu cektigi icin entube edilen Mehmet Ali
    Erbil, tedavisinin ardindan taburcu edildi. Cikista
    gazetecilerin sorularini yanitlayan Mehmet Ali, Ben Ingiltere
    'den gelen virusu gecirdim, cok agirdi. Gecirmeyenler ciddiye
     alsin dedi
Predicted summary:  koronavirusu yenen mehmet ali yilmaz
    koronavirus tedavisi goren yilmaz taburcu oldu
[{'rouge-1': {'f': 0.042553187904029274, 'p':
    0.09090909090909091, 'r': 0.027777777777777776}, 'rouge-2': {
    'f': 0.0, 'p': 0.0, 'r': 0.0}, 'rouge-l': {'f':
    0.04347825746691899, 'p': 0.1, 'r': 0.027777777777777776}}]
```

Code 4: Seq2Seq Results

As it can be seen, the result is interesting to say the least. The results will be talked about more in the comparison section. The skeleton of the code has been used here has been created by Madhav Mishra's Sequence2Sequence Article[Misb] and it also references the Bahdanau Attention Layer Implementation[Dzm].

6 Transformer

Transformer is a new state-of-art technology that handles long term dependency related problems for NLP and many more fields. Without having to use a architecture like RNN or LSTM, they use self attention technology mechanism to process inputs and produce outputs. they are considered to be more successful compared to other models in terms of handling long term dependency problems. Self attention mechanism has been considered as a small revelation in that regard. The research about the default Transformer model is done upon reading Jay Alammar's work on Transformer models[Ala].

INPUT
Je suis étudiant
THE TRANSFORMER
OUTPUT
I am a student

Figure 16: Simplified Transformer

6.1 Self Attention

Self attention will help the model to focus on the parts of the sequence it thinks is important and try to relate these parts of sequences with each other. To make it simpler, self attention tries to make similar connections within the same sentences.

The self attention layer is on a much lesser complexity and produces better results compared to normal recurrent layers when the context of processing the given data. An attention process within Transformer is divided to three parts: Query, key and value vector. By using dot product, query and key values will be multiplied and afterwards, they will be put into softmax parameter to be multiplied by value vector. Lastly, the sum value, which will be the probability of the sequence aimed for, will be handed to the model to process.

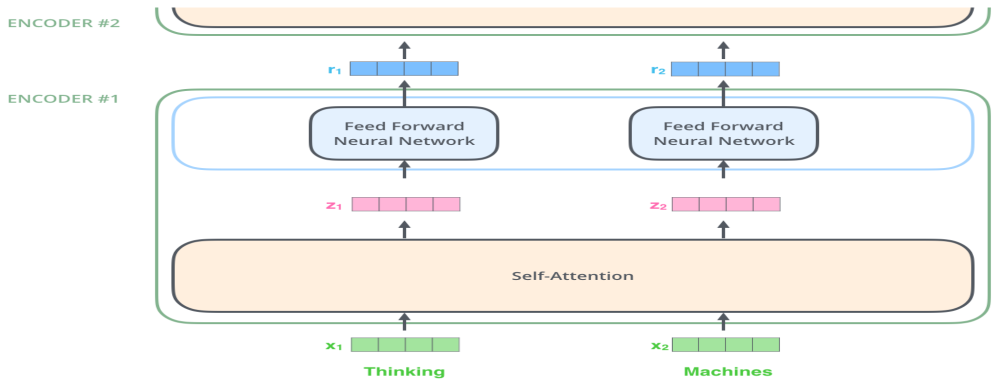


Figure 17: Attention Progress

6.2 Encoder-Decoder

Encoder and decoder processes the texts as the sequential vectors and produces a vector that will be the result of a prediction based on the processed knowledge. Positional encoding is used to create a sequence that will be pointed by the positions that have been showed by the in-out pointers. The Transformer's architecture does not contain any recurrence or convolution and hence has no notion of word order. To overcome this, positional encoding will be used. The input sequence is put in decoder without a specialized order or any specified instruction. It is important by the model to understand the meaning, the placement and the order of the text it is fed on.

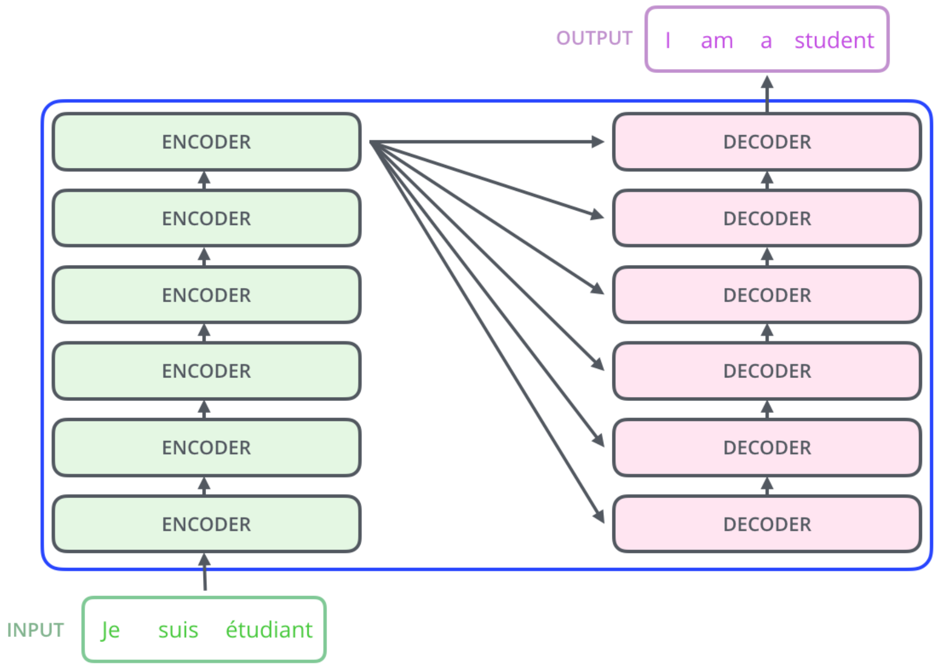


Figure 18: Encoder Decoder of T5

6.3 Multi Head Attention

Multi head attention is the situation where instead of using a singular attention mechanism, the usage of multiple attentions is at state. Every attention head has a different vectoral change within themselves. Transformer uses 8 different parallel or independent attention heads. With 8 different heads, 8 different versions of query, key and value set is used together. Multi headed attention layer serves its best purpose by making the results from its process interact with each other to find the best possible result.

7 T5

7.1 Basic Knowledge about T5

T5 is a pre-built transformer model that is suited for NLP related tasks. It works with the text-to-text principle which transforms a text input to a text output. This principle can be seen as similar to the Seq2Seq NLP projects which take sequence of words to predict the output. Text-to-text principle is different compared to Seq2Seq models in the way that it takes a sub-sequence of words to predict sub-sequence of output. This, in summarization terms, means that the output will be a paraphrased version of the input text, with unfocused parts staying the same after the prediction. The prediction can also use Transfer Learning, using already existing data with similarly related task, to predict new values from different inputs or from different datasets.

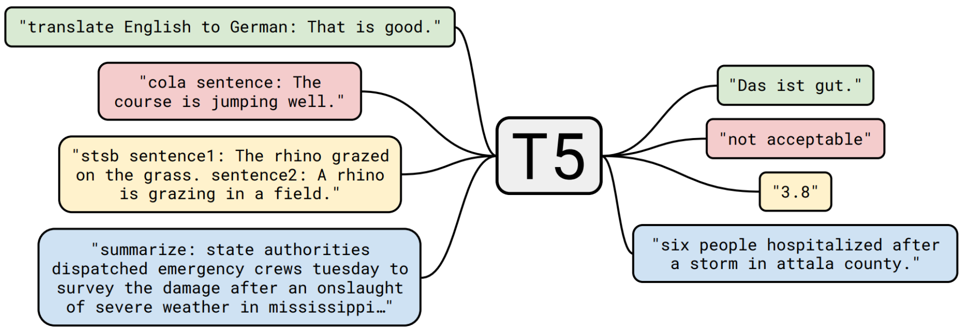

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 19: T5 Main Idea

7.2 Fine Tuning the T5

Fine tuning in Transformer models is a way to feed the dataframe that user wants to make experiments on to a pre-trained Transformer model. By giving the model labels with logical pointers, a suitable model would be able to predict the wanted results basing on the values it was fed on. Because the model is pre-trained, the data that is needed to be fed on is significantly smaller compared to a model like Seq2Seq.

Since this project focuses on summarization and T5 was chosen for this job. It was only fair to fine tune it for the summarization purposes. The data is read with pandas and tokenized by T5's own tokenizer. This process will be followed by deciding source and target sequences and padding their vectors to the max length for model to process it. The data will be split into train and validation and it will be loaded to the model using dataloader. This implementation took ideas from Abhishek Kumar Mishra work on T5 summarization[Misa].

Training will be under GPU powered CUDA to accomodate the power it needs to be completed. The model will try to predict the text sequences with the restriction made by self attention masking to get the best possible result. After the training is over, a predictions dataframe will appear for the test prediction results. Loss of the data will be provided using wandb, which holds the weights and biases of the Transformer model.

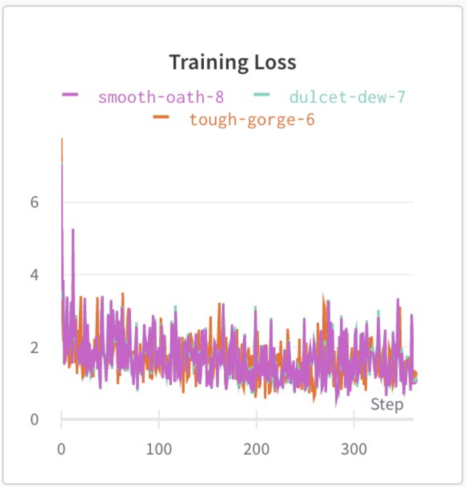


Figure 20: Loss Value for English Data

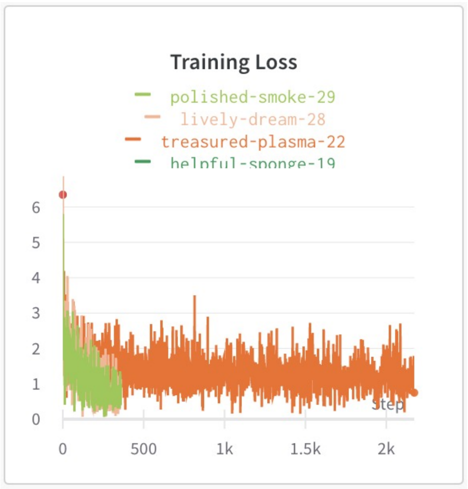


Figure 21: Loss Value for Turkish Data

The predictions dataframe can be used to make predictions that will implement same principles that the already given data had. Here is an example from the scrapped Turkish data.

```
#Testing
summarize("""Galatasaray'da tarihi secime sayili gunler kala
    eski baskan Unal Aysal'dan carpici bir hamle geldi.Bircok
    baskan adayinin cikmasi sonrasi Aysal, Adnan Ozturk ve Sedat
    Dogan'inda katilacagi bir toplanti yapma karari aldi.Burak
    Elmas ve E ref Hamamcioglu'nun da davet edilecegi toplanti
    da Aysal'in adaylara birlesme teklifi yapmasi bekleniyor.""")
>>>baskan adaynn ckmas sonras Aysal, Burak Elmas ve Sedat Doan'
    nda katlaca bir toplant yapma karar ald. Galatasaray'da
    tarihi secime sayl gunler kala eski baskan carpc bir hamle
    geldi.
```

Code 5: T5 Results

8 Measurement Tools

There will be a need to compare these algorithms with both themselves and different algorithms that do the similar summarization job. Of course, a simple look at the overall summarization of these different algorithms can create an overall image about their performance levels. Then again, a representation using something more concrete would always help more than it would hurt. Within Natural Language Processing, the summarization, like translation, is measured by two important values: BLEU and ROUGE score. With these values, some ideas and facts can be stated about both the model and their responsiveness on the data it was fed on.

8.1 BLEU Score

Bilingual Evaluation Understudy, or in short BLEU, is a score that although at first created for translation, it has known to be a good way to measure a NLP task. It can be used to evaluate the generated summary compared to the original one. The calculation is done by comparing the n grams of summary to the n grams of the original one, where 1-gram represent tokens of each word and bigrams that represent each pair of word. The best score possible is 1 out of 1 which, if the sentence is not completely the same, is impossible. This also means that while it is good in terms of finding the closest sentence, it is also flawed in the way of when similar word that give the same meaning to the sentence are a part of different summaries, BLEU will ignore this because the words is not the same. BLEU score is calculated using the reference of this[Bro] article

8.2 Rouge Score

Recall-Oriented Understudy for Gisting Evaluation, also known as Rouge, is a group of metrics that can be used to measure the legitimacy of the nlp task given. It works similarly compared to bleu but the main difference is bleu looks for how much of the words that were generated were in the original one whereas Rouge looks for how much of the words that were a part of the original one is a part of the new one. There are three main metrics that will be used in Rouge score: Rouge 1 and Rouge 2, which looks for the n grams of the two examples and Rouge L which looks for the longest same sequence possible. Rather than a single score, Rouge is measured by F1, precision and recall. The calculations are done according to the tutorials James Briggs[Bri] have developed.

9 Comparison

The main reason this project has been initiated is to observe the differences between these algorithms that, in theory, do the same job. The hard part was to classify these algorithms in terms of their strengths and their weaknesses within the situational moments and such. In this section, the algorithms will be talked about under these labels and try to get the most out of them from the experiences earned while conducting different experiments. The algorithms will be compared as singular functions at first, ending with comparing all of them in an measuremental manner.

9.1 TextRank

The main thing that observed about TextRank is that it can return a somewhat meaningful summary no matter how long the text really is. Since the PageRank algorithm will rank the most important sentences with their word vector values, the output will be one of the first main ideas that one can get from the text, even if it isn't the first one. This fact makes TextRank stand tall compared to other algorithms who need a fixed vector length for their datas to be processed.

One of the other benefits of TextRank algorithm is that it is basically cost free in terms of both data and required power from the backend of the system. It runs in a minimal nested system that its power and time cost can be ignored nearly all of the time. The algorithm's no need for a predefined vocabulary and its simplistic nature makes it maybe one of the most accessible and dependable summarization methods out there.

It is also bilingual, meaning any language based text can be used. Since TextRank is only interested in the numerical similarity value of the words, it won't recognize the language, rather it will simply ignore it. Compared to most models which require a specific language based tokenizers and data, the flexibility that TextRank offers is actually incredible.

Of course, It isn't perfect. TextRank algorithm depends on the words continuously citating each other within the text. This is the main reason why mainly news should be used to be summarized with this method, as they already cite the important parts within themselves multiple times. The problem comes in when the text isn't structured like a news and more like a story. When this happens, the output will not reflect any kind of the summarized version of the given text. The stories tend to not use the same words, other than the protagonists and such, too much, so it is understandable that TextRank will not be able to rank the sentences properly, as they won't be too distinguishable compared to each other.

9.2 Sequence2Sequence

Seq2Seq summarization model is quite the complex one to analyze. In terms of abstractive summarization methods, no other example really catches the meaning abstractive as well as Seq2Seq. With its decoder nature trying to predict sequences by the probability of tokens it specified the vector with, Seq2Seq can be seen as a child trying to associate words using their already defined vocabulary to create a meaningful answer within their own scope. Seq2Seq won't be always correct or sensible, just like any other variation of a deep learning method, but with the right approach, it will produce the most desired result set up by the user. Meaning, in theory, it can summarize any pattern, length or base it on any language as long as the methods are right and the labels and data fed on.

As pointed out in the last paragraph, Seq2Seq can learn any patterns within the language it is fed on as long as it is fed on right. Tokenizers that have been used are not language sensitive so they will be processed without a problem. If looked upon some results within the notebooks of Seq2Seq, whichever language is used, produced sensible results most of the time according to the basis of the language it is fed on.

One of the most important thing that needs to be talked about is the attention layer. As it can be seen in the notebooks the difference between with or without attention is sizeable enough to notice just by eye test. Although it doesn't guarantee it, Bahdanau's attention layer created for NLP purposes makes a great job of finding the important word to capitalize on when building a sequence, whereas no attention layer can be pretty distruptive to the main meaning when it tries to summarize it. With using 1000 data each, a comparison made between these two models using BLEU and rouge. There is a 5 percent difference in BLEU and as much as 10 percent difference in rouge values, with both made by attention-layered Seq2Seq. This result might have been a differen one, but in most samples, attention powered Seq2Seq is expected to come out on top, regardless of how much one values these prediction scores.

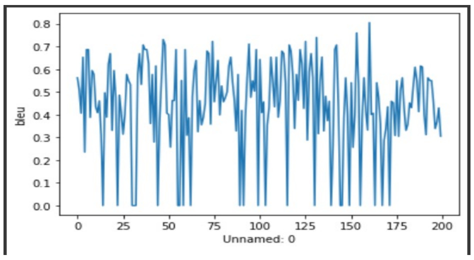


Figure 22: Seq2SeqW Attention

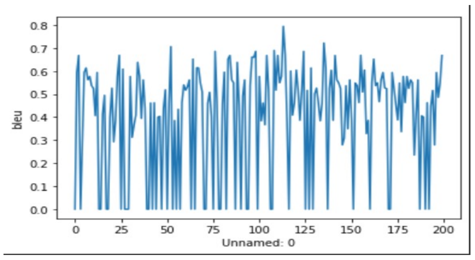


Figure 23: Seq2SeqWO Attention

The data is absolutely the most important part of this model. What processed by the model can either make or break the entire progress. In the case of Seq2Seq, it really is a matter of the bigger and more organized the data, the better. This basically means that data should be partitioned into what the model wants to operate on. If it wants to summarize 200 words into 20 word max sentences, then the data it should fed on must include, for the most parts, a sizeable amount of pairing similar to the aim and some anomalies included for the good measure. Of course, a model like this won't be much of a help when it goes againts its limitations. For example, it will have a hard time summarizing a text with the length of 70 words because it is not used to it. The researches and experiments done for this project showed that there must be a balance between the lengths of the texts and their sample size to create a all summarizing method. This project tried to use such data but even with 120.000-ish data it had trouble rationalizing some decisions it made when summarizing. There isn't an estimation about how big a data should be but it must be said that it should be bigger than 120.000, at least that is what is gotten from the experiments.

One other thing about the data is the importance of its variation. Softmax within dense layer tries to predict the values for the sequence according to the previous sequences the aimed word was used on. If the data is diverse and not big enough, the predicted outcomes won't be effective as the model will respond with lesser associated words to the focused sequence because it doesn't have enough instances to cover more topics. For example, one of the models trained by a partitioned version of the Turkish data always answered the summarization about Galatasaray with Fenerbahçe personalities. This means that there isn't enough variation on the topic of Galatasaray itself that it focuses on the news involving both of these teams. There are two solutions for this problem: Either manipulating the data to respond one particular topic perfectly or increasing the variation size, by that the overall size of the data, to predict from more previous instances.

The main negative, aside from the wrong predictions that are more of the data's fault, is that Seq2Seq is limited hardware wise when it comes to both memory and time concerned. From using Google Colab's own GPU, even it cannot allocate memory to summarize texts that are longer than 250 words. If that problem is solved, the time it needs to be trained is still quite much. This limits the capabilities of Seq2Seq greatly when it comes to summarization, as it won't be able to summarize most long texts if it isn't trained under a super computer and a sizeable amount of data. It is costly and it pretty much made Google penalize one of the authors(Tuğra) for using GPU for long periods of time.

9.3 Transformer

Transformer models are the latest and most advanced state-of-art technologies that text summarization is currently using. They are, as defined before, pre trained models that are waiting to be fine tuned for the purpose that the prepared data and labels instruct them upon. Mainly for this, they are quite similar to Seq2Seq models as they are both use a sequential way to create a word vector with them trying to build the aimed end sequence by using focused bits that they choose according to their algorithms. The main difference between them, however, is that Seq2Seq prediction progress takes the sequence word by word, whereas Transformer models take text to text approach, meaning they will look for focused meaningful parts as a whole rather than a word by itself.

Text to text approach also means that the abstractive nature of T5, the Transformer model that have been used in this project, will look different when the result comes in. This largely means that T5 will take parts of the sentence, paraphrase them into its own logic and change the importance order by the metric it thinks is more important. This also is a sign that by not predicting the whole word sequence one by one, Transformer models are more suited for the longer summarization processes.

One of the most important weapons Transformer models have in their arsenal is self attention, which will focus on entities(person, place, time etc.) and use them directly within the output sequence. This technology makes prediction sequences, while similar to one another, really accurate in terms of getting the grasp of the specific event. It will paraphrase the sentence is really the fastest way to explain this process.

The drawback that prevents T5, or most transformer models, to reach the undisputed pinnacle is that its tokenizer does not recognize foreing letters. Meaning even though the result won't change since the data is processed nonetheless, it won't look as good as, like, Seq2Seq. Some of the examples for this situation can be seen within the project notebooks for T5. If one goes around this problem for their model, it will certainly improve the experience.

As it was in Seq2Seq, the variation of data is also as important as for T5. Like Seq2Seq, it is imperial that the data is evenly distributed in terms of their category. Otherwise the results would simply be bad.

The main drawback of Transformer models is that they can be considered a bit slow when compared to other models. This is likely because of many functions that needs to run behind to create a prediction sequence. While it would be more precise in terms of meaning, It is not the best model for a mass produced summarization platform when the time of the customer is one of the most important elements.

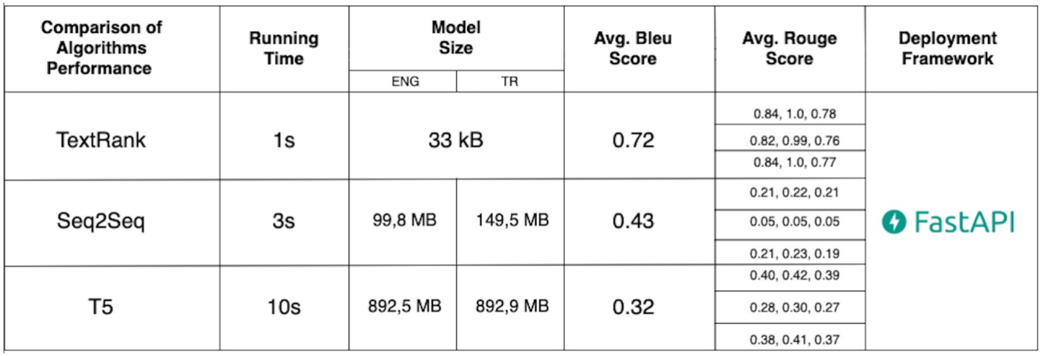
Models	Bleu	ROUGE1	ROUGE2	ROUGEL
TextRank	0.72	0.84,1.0,0.78	0.82, 0.99, 0.76	0.84, 1.0, 0.77
Seq2SeqATT	0.43	0.21, 0.22, 0.21	0.05, 0.05, 0.05	0.21, 0.23, 0.19
Seq2SeqNOATT	0.37	0.13,0.14,0.13	0.02,0.02,0.02	0.13,0.14,0.12
T5	0.32	0.40 0.42,0.39	0.28,0.30,0.27	0.38,0.41,0.37

9.4 Comparison of All

With all of them having a similar data to validate, around 1000 each, these algorithms will be compared using BLEU score and rouge score values. Rogue values represent as f1, precision and recall

As it can be seen, TextRank gave the best results out of them because of the sole reason of it already had the sentence it was looking to produce in the first place. Seq2Seq with attention gave the second best in terms of blue but when considered rouge scores too, both attention Seq2Seq and T5 model can be considered in a similar place. Seq2Seq appears to have more n-grams can be discovered within the original one whereas the original text has more n-grams that are the same with the T5 model, hence the difference between their BLEU and rouge scores. Although Seq2Seq without attention has a good BLEU score, its rouge score is quite bad. Lastly, the reason why T5 has a lower BLEU score compared to Seq2Seq is that when it is paraphrasing, it breaks its n-grams that could match with the original one, hence the lower score.

These values can be improved by many factors, much more with a bigger validation set, better dataset and many more factors that can improve these models that have talked about in their respective comparison place.



Comparison of Algorithms Performance	Running Time	Model Size		Avg. Bleu Score	Avg. Rouge Score	Deployment Framework
		ENG	TR			
TextRank	1s	33 kB		0.72	0.84, 1.0, 0.78	FastAPI
					0.82, 0.99, 0.76	
					0.84, 1.0, 0.77	
Seq2Seq	3s	99,8 MB	149,5 MB	0.43	0.21, 0.22, 0.21	
					0.05, 0.05, 0.05	
					0.21, 0.23, 0.19	
T5	10s	892,5 MB	892,9 MB	0.32	0.40, 0.42, 0.39	
					0.28, 0.30, 0.27	
					0.38, 0.41, 0.37	

Figure 24: Overall Performance Comparison

Speed-wise TextRank and Seq2Seq seem to be far more superior to the T5. Same relation can also be said about the spaces they occupy. The reason behind this is that pre-trained models occupy a lot of space so it is expected.

Data-wise, TextRank doesn't need to have any amount of data other that the one it wants to summarize. Since T5 is pre-trained, it requires a lot less data compared to Seq2Seq, which requires an absurd amount of data to reach the optimum states.

Length of the data-wise, Textrank will always perform better but there will be some problems by is being extractive, thus not being as effective pure summarization by itself. Both Seq2Seq and Transformer need a fixed length of data to summarize but they can do their job more elegantly compared to TextRank, with right data.

Training-wise, TextRank does not go into training since it is a simple function. An average training time that T5 requires seems to be around longer for the sampled data size it is based on compared to Seq2Seq. For instance, T5 will need 4 hours to process a data that has 5000 length in it, whereas Seq2Seq can go up to 70000-80000 samples within the given timeframe, if the power is there.

All in all, TextRank gives an idea of a text summarization method in a more basic but effective way. It will be hard for it to tackle complex and story like text though. Both Seq2Seq and Transformer have their optimum cases and their accuracy will surely better that TextRank just because they look for the meaning rather than word references.

10 Deployment

After the optimum models are saved and thought to be ready, they are deployed to be used by user inputs. With saving encoder and decoder models using keras and pytorch, models will be functionalized to be used within an html input. The basic steps are to setup an environment, load the models to python backend and load them to a web application with the usage of FastAPI.

10.1 FastAPI

FastAPI is a web framework that, like Flask, offers a variety of different solutions for any number of web applications. Like both Flask and RestAPI, FastAPI uses methods like get and post to connect web application part to the backend. With the help of uvicorn and jinja2 templates, FastAPI offers easy and fast services to create a more idealized version of a python based app.

10.2 Environment

A default environment is created in order to populate the needed libraries for the app to launch. Only the extensions of the libraries which are used in the code will be included. Since the defined libraries and their versions are compatible with pretty much any other system, users will have the choice the implement only the important libraries to keep their allocated memory as big as possible, meaning more memory to perform for the main app. Making the app as fast as possible is the reason why an environment is mainly created from the creator's standpoint. This can also be interpreted from the implementer's viewpoint as finding only the needed libraries would become much easier for them. Although the default conda environment also pretty much has all the libraries needed, the unneeded ones are expected to make the application slower. All in all, it is a must to have a customized environment for the app to go live.

10.3 Design of the App

Other than its style, which can be seen in the figure, the design of the app will include three basic parts that will serve as a bridge between summarizer functions and app itself. First, a dropdown list to choose which model will be used. Secondly, the textarea that the original text will go. Lastly, the invisible part under text area that will include summarized version once summarization is complete. These informations will be shared using post and get form methods of FastAPI. Improvements will be made to the style of overall app but function wise, it is thought to be a success.

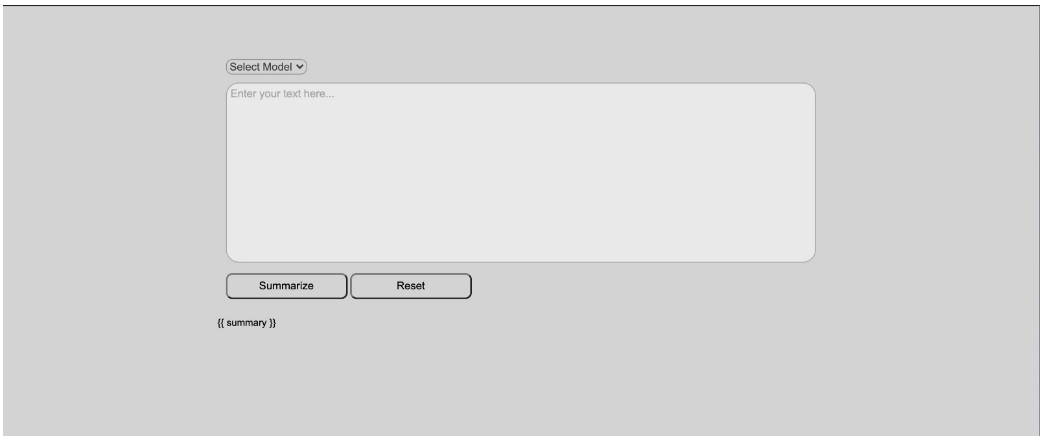


Figure 25: The General Look of the App

11 Discussion

The subjects are now even more famılliar to the authors, it was only needed to improve them with both different data and approach. After the experiments finished, it can be said that the overall results using these algorithms can be considered a success. Even though there is still so much to do, the progress and their deployment has made this project worthwhile.

One of the main topics that have been wanted to be implemented for this project was to use NER in a summarization method. Although it has been tried for the last two months for both TextRank and Transformer, the implementation was a failure and overall needs more time and experience to be implemented.

11.1 Risk management

The risks that have been considered were mostly about time constraints and most of them unfortunately came true. With a project as big as this, that is to be considered sadly. Thankfully they weren't that much of a problem for completing the project.

12 Conclusion

From the project's part, it has been concluded that all 3 of these algorithms have their strengths and their weaknesses. Comparison section should be helpful to the people who are still a bit far away from the processes of these algorithms. Although there are still many experiments to be done and much more data to be pulled, this project has achieved most of the points that it set onto. All 3 models have been known better, Turkish language has been implemented for all of them and FastAPI deployment has also been achieved. Although some things may have been done better, this has been satisfying for what it is.

Obviously there are still improvements to be made for nearly all of the models, mainly because of the data, but the path to finding an optimum summarization model must go on. Until the end result is reached, this project will need to satisfy the needs.

12.1 Realistic Constraints

This was a subject that was a bit far away from the ideals that this project was set on. Ultimately, the cost of the memory and time when training these kinds of high tech models will always be more than any normal system would be able to handle. Even though Google Colab's resources were used, they were also limited in time, specifically 8 hours in 1 day and the time that it would be able to be used got even lesser each time GPU was run throughout its designated time.

Data was also a problem in terms of availability. Even though data was scrapped for this project, it still wasn't enough to get a good partition in to get the ultimate model out of whatever model that was being used. With more reseach about the data and resources, this project can easily get better in all aspects that it lacks. Sadly, some constrains are far too much to beat within the designated time.

12.2 Social, Environmental and Economic Impact

Summarization, although being a popular task among NLP crowd, hasn't seem to be compared as extensively as this paper does. Although anything on this paper might not be enough for people to decide which model should they use in a straight up answer because of its relatively small sample size(around 110.000 data for both languages, still not enough) and impressions, it should give an idea about the models' performance levels on different situations. Which ones are good at long texts, which ones gives better result for designated values, which ones are easy to implement are only some of the questions that this paper tries to answer. If this project can make a person's quest on using these algorithms easier, it can be called a success.

12.3 Cost Analysis

Time was the most important part to plan in this project. Since Google's GPU can only be used 8 hours a day, it was crucial that the training should not go for too long, even though training without overfitting as long as possible would have made the optimal values for the predictions. With trying to fit at least 4 training sessions within a week, which makes in minimum 32 hours a week semi-supervised, for at least 4 months. Even when not training, most of the other remaining time went onto data scrapping and even more research about the possibilities that these models can handle. The average amount that a software engineer that responsible from research and development fluctuates between 4000-5000 TL. It would be a fair amount of compensation if salary was something along those lines.

12.4 Standards

This project is believed to be not breaking any engineering code of conduct with no intention of making any profit out of the implemented functions that designed by somebody else nor using the scraped data anything but the main training. It has been developed by nothing but experimental value and only holds that value alone.

Although there wasn't a specific standard for this project to follow on, P2841 - Framework and Process for Deep Learning Evaluation seems to be the closest one since both Seq2Seq and Transformer were trained using deep learning approaches.

12.5 Contributions

We would like to thank our teacher Uzay Çetin for instructing us whenever he can to help us in our static phases. We also would like thank nice people over Google Colab for letting us use their GPU when we needed it. Lastly, we appreciate the insturcions and models in HuggingFace[Fac] that got us through Transformer where there isn't a concrete reference.

References

References

[Ala] Jay Alammar. *The Illustrated Transformer.* URL: https://jalammar.github.io/illustrated-transformer/. (Accessed 1 June 2021)).

[Bri] James Briggs. *The Ultimate Performance Metric in NLP.* URL: https://towardsdatascience.com/the-ultimate-performance-metric-in-nlp-111df6c64460. (Accessed 1 June 2021).

[Bro] Jason Brownlee. *A Gentle Introduction to Calculating the BLEU Score for Text in Python.* URL: https://machinelearningmastery.com/calculate-bleu-score-for-text-python/. (Accessed 1 June 2021).

[Dzm] Yoshua Bengio Dzmitry Bahdanau Kyunghyun Cho. *Neural Machine Translation by Jointly Learning to Align and Translate.* URL: https://arxiv.org/pdf/1409.0473.pdf. (Accessed 1 June 2021).

[Fac] Hugging Face. *Transformers.* URL: https://huggingface.co/transformers/.

[Haj] Nissan Hajaj. *Producing a ranking for pages using distances in a web-link graph.* URL: https://patents.google.com/patent/US9165040B1/en. (Accessed 1 June 2021).

[Misa] Abhishek Kumar Mishra. *Fine Tuning Transformer for Summary Generation.* URL: https://github.com/abhimishra91/transformers-tutorials/blob/master/transformers_summarization_wandb.ipynb. (Accessed 1 June 2021).

[Misb] 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 1 January 2021).