THE REPUBLIC OF TURKEY

BAHCESEHIR UNIVERSITY

AN EVALUATION OF AUTOMATIC TEXT SUMMARIZATION TECHNIQUES

Master's Thesis

MURAT GÜMÜŞ

İSTANBUL, 2019

THE REPUBLIC OF TURKEY

BAHCESEHIR UNIVERSITY

INSTITUTE OF SCIENCE

COMPUTER ENGINEERING

AN EVALUATION OFAUTOMATIC TEXT SUMMARIZATION TECHNIQUES

Master's Thesis

MURAT GÜMÜŞ

Thesis Supervisor: ASSIST. PROF. DR. TEVFİK AYTEKİN

İSTANBUL, 2019

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES COMPUTER ENGINEERING

Name of the thesis: An Evaluation Of Automatic Text Summarization Techniques
Name/Last Name of the Student: Murat Gümüş
Date of the Defense of Thesis: 29.05.2019

The thesis has been approved by the Graduate School of Natural and Applied Sciences.

Assist. Prof. Yücel Batu SALMAN
Graduate School Director

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Sciences.

Assist. Prof. Tarkan AYDIN
Program Coordinator

This is to certify that we have read this thesis and we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Sciences.

Examining Comittee Members	Signature
Thesis Supervisor	
Assist. Prof. Dr. Tevfik AYTEKİN	……..……..
Member	
Prof. Dr. Mehmet Alper TUNGA	……..……..
Member	
Prof. Dr. Yücel SAYGIN	……..……..

ABSTRACT

AN EVALUATION OF AUTOMATIC TEXT SUMMARIZATION TECHNIQUES

Murat GÜMÜŞ

Computer Engineering

Thesis Supervisor: Assist. Prof. Dr. Tevfik AYTEKIN

May 2019, 69 pages

From the beginning of mankind's existence, the series of meaningful voices produced and the rules in which these sequences are written are called natural language. Language is the most important communication protocol that representation of human intelligence. In order to operate this protocol, quite complex transactions are taking place at the neocortex part of our brain. Even though Neocortex is very successful in processing the data obtains, the increasing data density makes it difficult to find the meaningful part within the data. It takes so much time to find important parts from the heaps of data. The answer to the question of how we can reach meaningful information in a short time is text summarization. Simply, text summarization is the process of shortening long texts by using natural language (NLP) techniques. There is various application areas of summarization processes, for instance article summarization or making weather forecast systems, news and financial data readable.

In the past, many summarizing practices were made using extractive methods. Extractive text summarization techniques are based on the selection of existing sentences within the text. Successful solutions can be applied using extractive methods. But humans can do more than extractive techniques for summarizing. When we try to summarize a text, our brain forms the semantic representation of the text. We are summarizing from this representation. This is called the abstractive method.

In this thesis, summarizing text using natural language processing, extractive and abstractive methods are investigated. Differences between methods are being evaluated and interpreted.

Keywords: Text Summarization Techniques, Deep Learning, Language Modelling, Natural Language Processing

ÖZET

OTOMATİK METİN ÖZETLEME TEKNİKLERİNİN DEĞERLENDİRİLMESİ

Murat GÜMÜŞ

Bilgisayar Mühendisliği

Tez Danışmanı: Dr. Öğr. Üyesi Tevfik AYTEKIN

Mayıs 2019, 69 sayfa

İnsanoğlunun varoluşundan itibaren ürettiği anlamlı sesler dizisi ve bu dizilerin yazılı olarak ifade edildiği kurallar bütününe doğal dil denir. Dil insan zekasının göstergesi olan en önemli iletişim protokolüdür. Bu protokolü işletebilmek için beynimizin neocortex kısmında oldukça karmaşık işlemler gerçekleşir. Neocortex elde ettiği veriyi işlemede oldukça başarılı olsa da günden güne artan veri yoğunluğu veri içinden anlamlı kısmı bulup çıkarmasını güçleştiriyor. Veri yığınları arasından önemli kısımları bulmak oldukça zaman alıyor. Bize lazım olan anlamlı bilgiye kısa zamanda nasıl ulaşabiliriz sorusunun cevabı metin özetlemedir. Basitçe, verilen uzun bir metinden doğal dil işleme (NLP) tekniklerini kullanarak metni temsil eden anlamlı bir özet çıkarma işlemine metin özetleme denir. Özet oluşturma işleminin birçok uygulama alanı bulunmakta, örneğin makale özetleme, hava durumu tahmin sistemleri, haber, finans verilerinin okunabilir hale getirilmesi gibi.

Geçmişte birçok özetleme uygulaması çıkarım yöntemleri kullanılarak yapılmaktaydı. Çıkarım metin özetleme teknikleri, metin içinden var olan cümlelerin seçilmesine dayanıyor. Çıkarım yöntemleri kullanılarak başarılı çözümler uygulanabiliyor. Fakat insanlar özetleme konusunda çıkarım özetleme tekniklerinden çok daha fazlasını yapabiliyor. Bir metni özetlemeye çalıştığımızda beynimiz metnin semantik gösterimini oluşturuyor. Bu gösterimden ise özet çıkartıyoruz. Buna soyut eğilimli yöntem diyoruz.

Bu tez çalışmasında doğal dil işleme, çıkarım ve soyut eğilimli yöntemler kullanarak metin özetleme işleminin nasıl gerçekleştiği araştırılmıştır. Yöntemler arasında farklar değerlendirilerek yorumlanmaktadır.

Anahtar kelimeler: Metin Özetleme Teknikleri, Derin Öğrenme, Dil Modelleme, Doğal Dil İşleme

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

BLEU	:	Bilingual Evaluation Understudy
BOW	:	Bag Of Words
BRNN	:	Bidirectional Recurrent Neural Network
CBOW	:	Continous Bag Of Words
CNN	:	Convolutional Neural Network
EOS	:	End Of Sentence
GLoVE	:	Global Vectors For Word Representation
GRU	:	Gated Recurrent Units
LSA	:	Latent Semantic Analysis
LSTM	:	Long Short Term Memory
MSE	:	Mean Squared Error
NLP	:	Natural Language Processing
PR	:	PageRank
RNN	:	Recurrent Neural Network
ROUGE	:	Recall Oriented Understudy for Gisting Evaluation
SEQ2SEQ	:	Sequence to Sequence
SNARC	:	Schotastic Neural Analog Reinforcement Computer
SVD	:	Singular Valued Decomposition
TF-IDF	:	Term Frequency-Inverse Document Frequency
UNK	:	Unknown

1. INTRODUCTION

Alan Turing begins his article **Computing Machinery and Intelligence** in 1950, with a question "Can machines think?" In order to be able to answer this difficult question, he says that it is possible to define of the words '**machine'** and '**think'**. Turing wander away from the real meaning of these words, asserts that different results will be achieved and instead of the question "can machines think?" introduces a game called **Imitation Game**. **(Turing et al., 1950)**

There are one woman, one man and gender not trivial interrogator in imitation game. The interrogator's task is to find the gender of the male and female participants who are unidentified. The interrogator may ask questions to other participants for this purpose. One of the participants helps the interrogator and the other tries to mislead. Interrogator says its answer at the end of the game. How would respond if there was a machine instead of interrogator? Turing (1950, pg. 7) replies the question:

> *I believe that in about fifty years' time it will be possible, to programme computers, with a storage capacity of about 10^9, to make them play the imitation game so well that an average interrogator will not have more than 70 per cent chance of making the right identification after five minutes of questioning.*

Nowadays we see that time justifies Turing. Although this is an important question that we have come to this point in time, it will shed light on the subject of text summarization.

If we continue our journey of time with machines and intelligence. **Marvin Minsky** and **Dean Edmonds** made a computer called **SNARC (Stochastic Neural Analog Reinforcement Computer)** in 1951, simulating the first artificial neural network. SNARC had synapses that set weights according to the success of the task. Marvin Minksy argues that the human mind consists of small pieces called "agents"', but that these parts do not have their minds in his book **Society of Minds**. In 1956, the term artificial intelligence was first used by **John Mccarthy**. **Frank Rosenblatt** has been developed base of modern neural networks. He described the perceptron, the artificial model of a neuron, as he emphasized in the title of "**the perceptron: a probabilistic model for information storage and organization in the brain"**, which published in 1958. He says 3 questions should be answered in order to model a neuron like in a biological system. The first question is how the information is defined, found and felt in a biological system. The second question is about how the information is remembered and stored. The last question concerns how stored information affects behavior. **Frank Rosenblatt** elaborates the perceptron by excluding the first question that makes the difference between man and machine**. (Rosenblatt et al., 1958)**

Along with the artificial intelligence studies, in the 1950s, with the development of computers, progress was made in many areas. One of them was the linguistic area.

Luhn Algorithm, developed by Hans Peter Luhn in 1958, is the first study in the field of text summarization. In 1960s, the advances in artificial intelligence research were not enough in the field of application that's why the financial support for the researches decreased. This period of stagnation continued in the 1980s until the old ideas were rediscovered by new approaches such as machine learning. The most important development in the 1980s was the backprogation algorithm, which allows training machine published by Geoffrey Hinton, David Rumelhart and Ronald Williams **(Hinton et al., 1986).** The main turning point for artificial intelligence was IBM's Deep Blue, in 1997 defeating world chess champion Garry Kasparov. However, in 1985 he had the chance to beat up 32 computers in a tournament in Hamburg. Garry Kasparov won the first game in 1996. With this defeat, the golden age of human machine interaction has begun. One of the biggest differences between artificial intelligence applications and artificial intelligence applications of 1997 is data. In other words, machines can be trained on large data stacks. After 2012, the machine learning discipline, which is the sub-discipline of the artificial intelligence domain, gained momentum again due to the old ideas that were found in the 1950s. Nowadays, artificial neural networks are finding solutions many problems, promises great future.

One reason why artificial neural networks are so popular is that Deep Blue, which defeats Kasparov, has more processing power, so it calculates more moves and makes fewer mistakes, but these qualities do not make a machine smarter than a human. We can think of a car as faster than a horse, as it doesn't make the car more intelligent than a horse. This problem has led to neurons being searched for information and methods of transmitting information to another neuron. Data which is learned by neurons from input and output information, enables that learn from previous errors. This is called training the machine on a problem. In addition, the use of artificial neural networks with parallel programming techniques has increased considerably. By this way, the ability to solve complex problems, to choose a feature from a picture **(Convolutional Neural Network)**, to able to create a sentence by using their memory **(Recurrent Neural Network)**.

Today, these methods are used in many areas. One of these areas of application is my thesis subject **text summarization**.

1.1 AUTOMATIC TEXT SUMMARIZATION

Nowadays, with the introduction of technology in every field of human life, the amount of data around us is increasing rapidly in proportion to this development. Let's think we do research on a subject, we are encountring too much text-based material about this subject. It will be difficult to focus on what we are looking for in the data density. In time, our productivity will decrease. Maybe we're going to have missed an important part of a text. We encounter these situations every day. Similar problems make summarizing function a necessity.

Automatic text summarization is a meaningful streamlined short version of a long text.
In summary, a title related to a news or an article, trailer images in the films, book summaries, television broadcast stream summaries, summary information about the weather, the summary of financial information, history of the list of important events can be given as many examples.
How can we create a summary of a given text? In the literature, there are two different approaches about this question. Extractive methods and Abstractive methods. Text summarization can be classified as singular or multi document according to the input type, generic to its purpose, domain specific or query based, as output or as extractive or abstractive **(Kumar et al., 2016)**.
In the scope of the thesis, the input type multi-document text was converted to single document type. Generic type, ie a general type of text was summarized. Both abstractive and extractive summarizing techniques were mentioned as output type.

1.1.1 Extractive Summarization

In extractive summaries, words or sentences that are important in the text are selected. A text is created from the selected sentences. Let's think that we have highlighted the sentences in the book or notebook that we read while working for an exam. Extractive summarization is a process as in the example. Luhn’s algorithm, latent semantic analysis (LSA), graph based algorithms, textrank and lexrank were used in extractive summarization techniques.

1.1.2 Abstractive Summarization

Abstractive summaries are techniques based on producing a new text related to the given subject. Let's say we're trying to extract a summary of a topic, for this we read a long part of the text. Then we create a text from the words that remain in our minds. Abstractive summarization provides a summary of the human approach. Artificial neural networks can be used to summarize with abstractive summarization methods.Within the scope of the thesis, deep learning methods have been examined and summation studies have been done by using **recurrent neural network (RNN)**. The results of both methods will be evaluated with **BLEU** and **ROUGE** metrics.

2. THEORETICAL STUDIES

2.1 NATURAL LANGUAGE PROCESSING

Today, by development of social media, there is a high amount of data which is processed or waiting to be processed on the internet. The concept of natural language processing emerges at this point. Natural language processing (NLP) is a branch of artificial intelligence that aims to help the processing, using and manipulating of natural languages such as Turkish, English and German. Applications such as machine analysis, chatbot and text summarization include both artificial intelligence and text data processing. In order to solve the problems covered by natural language processing such as text summarization, we need to examine some basic nlp issues first. In the scope of the thesis, the python programming language and the nlp library nltk in python were used.

2.2 LANGUAGE MODELLING

2.2.1 Tokenizing Word and Sentences

Tokenization is a process of dividing a paragraph into sentences, sentences into words. In NLP analyzing text on the text called corpus. For example, 'Natural Language is branch of artifical intelligence.' When we divide the sentence into spaces, we get a token list of words. We can use tokenize function in nltk library to be able to tokenize the text on Corpus. One of the first things we need to do to work on the text is tokenizing. We can also use the split function, which allows us to divide sentences from spaces instead of the token function.

2.2.2 Stemming and Lemmatization

Stemming tries to reduce the derived words to the roots of words. For example, when you apply the stemming process to the words 'gone', 'goes' and 'going', we get the word 'go' as the word root. There is a problem when we apply stemming. For instance, the common roots of intelligent and intelligence words are intelligen. But the word intelligen seems to be not a word for us.The method that solves this problem is called lemmatization in NLP. Lemmatization reduces the words to word roots like stemming. In doing so, unlike stemming, the word meaning is preserved. For instance, if we have the words like intelligence, intelligently and intelligent as the word root. There are different situations in which both methods are used. For example, applying the stemming process on the text requires more processing power than the lemmatization process, but also takes less time. On the other hand, if the meaning of words is important in our text analysis, it would be appropriate to choose lemmatization method. For the Stemming process, we can use

the stem function of the PorterStemmer object in the Nltk library. Similarly, we can apply the word to the words using the lemmatize function of the WordNetLemmatizer class.

Source: (Standford Nlp, 2009)

2.2.3 Stop Words

Stop words are ineffective words extracted from text before doing analysis on a text. In general, the most commonly used words in that language, conjunctions, abbreviations are used as the words. For English, we can show examples such as 'a, about, at, are, is'. Similar to Turkish, words like “bir, birçok, kimi, kime, nerede, nereden, önce” are included in the stop words list.Nltk library stopwords.words (‘English’) function by providing the parameter English to get the stopwords list, and we use or omit them from our text.

2.2.4 Parts of Speech Tagging

Parts of speech tagging is the tagging process of text like of the verb, adjective or subject. We can do this by using pos_tag function in Nltk library.

2.2.5 Named Entity Recognition

In Corpus, we may want to find out the names, contact names, organization names or company names from the sentences. With the Named entity recognition methods, we can find certain words that pass in a sentence. In order to do a named entity recognition on the text, we first use the text pos_tag function to assign tags according to the elements. Then, using the ne_chunk function, identifies the named entity for the tagged word.

2.3 BUILDING LANGUAGE MODELS

In order to solve NLP problems such as text summarization, the algorithms that we will put into practice must work on numbers rather than words. We need to create a model for apply algorithm on that model. The model will represent the text, and we need to apply our algorithms on it. We can create models using different methods.

2.3.1 Bag of Words Model

The Bag of words model is one of the ways of displaying text data. Using machine learning algorithms on the model, we can make operations according to our purpose. For example, suppose we are trying to design a bag of words model through three sentences. Our sentences are as follows.

a. “I am going to school today.”
b. “I have to study math after school.”
c. “Math is going to be hard for me.”

The words in the sentence need to be converted to lowercase letters to eliminate case sensitivity. If we examine the sentences, we see that some words have passed more than once and some are used only once. **Table 2.1** indicates that frequency of words in sentences.

Table 2.1: Word counts

Word	to	i	going	school	study	math	am	today	have	after	is	be	hard	for	me
Count	3	2	2	2	2	2	1	1	1	1	1	1	1	1	1

We can see the words in the three sentences listed in the table sorted by the frequency of passing. In order to define the model of bag of words, we need to display the 6 most commonly used words and their frequency on a matrix. As shown below **Table 2.2**, we can form a matrix that shows the frequency of words by sentence.

Table 2.2: Bag of words model

Words/Sentences	to	i	going	school	study	math
1	1	1	1	1	0	0
2	1	1	0	1	1	1
3	1	0	1	0	0	1

The matrix that we create is a bag of words model representing our text. A value of 1 in the matrix indicates that the word is passed in the sentence and 0 is not in the sentence. Based on this model, we can use machine learning algorithms to analyze the text.

2.3.2 TF-IDF Model

Another model used in language modeling is the TF-IDF model **(Robertson et al., 2004).** Bag of words models we have examined before, we have two different problems. The first is about not keeping the word values on the model. If we examine the BOW matrix we created above, we see that all the words have the same value. In other words, the words in the sentences are represented by a value of 1. In this case, we can not keep the importance of importance between two different words.

When we apply any machine learning algorithm on the model using the BOW model, we think that the words have the same importance. TF-IDF solves this problem successfully. The second problem is that the semantic information is not stored. The meanings of the words in the BOW model are not represented. The TF-IDF model also not able to solve this problem. We can do

word embedding matrix for this problem later on. If we look at how the TF-IDF model was created.

TF-IDF, TF (Term Frequency) IDF (Inverse Document Frequency) is the product of two different calculations. In other words, **TF-IDF = TF * IDF** can be called. Term frequency defines the frequency of a word on the sentence in which it is used. The value of inverse document frequency defines the value of the word on all corpus. The product of TF and IDF values represents the TF-IDF of the word. The formula used for term frequency is as follows.

$$\mathbf{TF} = \frac{(\textbf{Number of occurance of a word in a sentence})}{(\textbf{Total number of words in sentence})} \qquad \textbf{(2.1)}$$

By using the formula **(2.1)**, if we try to calculate the term frequency values of the 3 given sentences.

Table 2.3: Term frequency table

Words/Sentences	to	i	going	school	study	math
1	0.167	0.167	0.167	0.167	0	0
2	0.142	0.142	0	0.142	0.142	0.142
3	0.125	0	0.125	0	0	0.125

If we examine the **Table 2.3** with sentences, the word “to” is passed once in the first sentence, and the total number of words in the first sentence is 6. TF = 1/6 = 0.167. Similarly, the word 'to' passes once in the second sentence, and we find the word 7 in the sentence and TF = 1/7 = 0.142 if we apply the TF formula. If we apply the term frequency formula for all words, we obtain the term frequency matrix as seen above. In the next step, we need to create the Inverse Document Frequency matrix. We calculate the IDF value with the following formula.

$$\mathbf{IDF} = \log\left(\frac{\textbf{Number of sentences}}{\textbf{Word occurance in all sentences}}\right) \qquad \textbf{(2.2)}$$

Table 2.4: Inverse document frequency table

Words	to	i	going	school	study	math
IDF	0	0.176	0.176	0.176	0.477	0.477

If we try to calculate equation **(2.2)**, the id value for the word “to”, there are 3 sentences in the corpus, and the word “to” exist in 3 sentences. If we write the values in the formula **(2.2)** IDF = log (3/3) = 0. Similarly, when we calculate the formula for the word 'i', IDF = log (3/2) = 0.176. When we use the formula **(2.2)** for all words, we create the IDF matrix which is shown as above **Table 2.4**.

In order to create the TF-IDF model, we need to multiply the TF and IDF matrices according to the formula.

Table 2.5: TF-IDF matrix

Words/Senten	to	i	going	school	study	math
Sentence 1	0	0,029	0,029	0,029	0	0
Sentence 2	0	0,024	0	0,024	0,067	0,067
Sentence 3	0	0	0,022	0	0	0,059

When we multiply the TF and IDF values, we obtain the TF-IDF matrix above **Table 2.5**. We can think of the resulting matrix as a language model. When we examine the generated language model, we can say that the words with TF-IDF value are high, are more important words. For example, the most important words in the sentences according to the model are "study" and "math". The most trivial word is seem as the word "to". Thus, we can say that the TF-IDF method can help us analyze the given text while working on larger corpus.

2.3.3 N-Gram Model

Another model created in natural language processing applications is N-Gram Model. The N-Gram model is based on the **Markov Chain** principle. In the literature, the example of the gambler usually plays a coin is given, for explain the Markov Chain principle. If we take the issue through the example, consider a gambler who have a coin. Each time a heads to the gambler wins 1 lira, each tail lose 1 lira. Let us think that the gambler has played 10 times and the gambler have 5 liras in 10 games. When the gambler wants to play the game for the 11th time, according to the principle, it is enough to know the current money. According to Markov Chain principle, it does not matter how much he wins or loses in the sections he has played so far 11. At the end of the game, the gambler's money will be either 4 liras or 6 liras. In this case, we can say that the previous cases does not make any difference to predict the situation.

In the N-Gram model, probability cases in head or tails game are defined by words, sentences and characters. The characters in the sentence can be considered as items included in the Markov Chain. In other words, we can say that the characters which are created or the sequence of the sentence do not have any effect on the word or character to be created after the characters. For example, if we take the following sentence.

"I am going to school today."

If N = 2, the model will be as follows.

"I "," a","am","m "," g","go","oi","in","ng"," etc

If N = 3, the model will be as follows.

"I a"," am","am ","m g"," go","goi","oin","ing" etc

Similarly, we can apply the same method in words as in characters. If we set N = 3, our model will be seen as below.

“I am going”,”am going to”,”going to school”,”to school today”

In Google search character based auto complete can be application example of N-Gram based language modeling.

2.4 EXTRACTIVE METHODS

2.4.1 Latent Semantic Analysis (LSA)

Latent semantic analysis is a technique that analyzes the relationship between the data sets contained in a text and the terms contained in the data sets in natural language processing. **(Landauer et al., 1997)** In other words, it classifies the subject or the concept of the sentences in the text. Latent semantic analysis is required to be a language model. Let's assume that our model has MxN matrix to indicate N phrases, N words we have created before. In fact, we need a lot more sentences to implement LSA.

Table 2.6: Input matrix

Words/Senten	to	i	going	school	study	math
Sentence 1	0	0,029	0,029	0,029	0	0
Sentence 2	0	0,024	0	0,024	0,067	0,067
Sentence 3	0	0	0,022	0	0	0,059

We can say that the language model we created as in the **Table 2.6** above is an input matrix. In order to reveal the concepts or topics in the given text, we have to do the **SVD (Singular Valued Decomposition)** calculation on the input matrix. The singular valued decomposition, factorize the given input matrix. If we examine the equation below.

$$A_{[mxn]} = U_{[uxr]} * S_{[rxr]} * (V_{[nxr]})^T \qquad (2.3)$$

A: Input Data Matrix mxn matrix (m number of sentences, n number of words)

U: Left Singular Matrix mxr matrix (m number of sentences, r number of concepts)

S: Rank Matrix (Diagonal matrix) rxr matrix (r = rank of A)

V: Right Singular Matrix nxr (n number of words, r number of concepts)

We express the language model as the product of 3 different matrices using the input matrix which shown as equation **(2.3)**. For text summarization, we define concepts according to the concept and word according to the sentence we use. After this stage, we need to make a sentence selection

using the results obtained after SVD. Thus, we can understand that it is basically 3 steps to make text summarization application by using Latent semantic analysis technique.

Step 1: Forming the input matrix, in this step bag of words or TF-IDF may be preferred.

Step 2: Calculating the singular valued decomposition (SVD) on the input matrix.

Step 3: Choosing the most appropriate sentence according to SVD results.

There are various methods in the literature to select the most appropriate sentence in Step3. (**Çiçekli et al., 2011)** The first of these methods is **Gong and Liu's Approach, (2001).** In the Gong and Liu method, the $\mathbf{V^T}$ method distinguishes between the sentences and concepts in the right singular matrix, in other words, choosing a sentence from the most important concept.

In the **Steinberger and Jezek (2004)** method, unlike the Gong and Liu approach, the $\mathbf{V^T}$ matrix as well as the **S** rank matrix are used. To determine the selected sentence, the length scales corresponding to each sentence line in the $\mathbf{V^T}$ matrix are checked. Length values are calculated using concepts less than or equal to the given dimension. S rank matrix is used to determine the most important matrix. In addition, Steinberger and Jezek (2004) method can be selected more than one sentence.

Another method, **Murray, Renals and Carletta's approach, (2005)** approach in the $\mathbf{V^T}$ and **S** matrix is used in the selection of sentences. Multiple sentences of the most important concepts can be selected. How many sentences to choose the S rank matrix is decided using.

Finally, **Ozsoy's approach, (2010)** contains two different methods. The first one is **Cross Method**, we can say that it is the expanded form of **Steinberger and Jezek (2004).** Here is the preprocessing step between the SVD calculation step and the sentence selection step. The purpose of the preprocessing is to try to eliminate non-significant sentences of the concept. For this, a mean value is calculated for each sentence. If the cell value of Concept and the sentence is less than or equal to the average, it is equal to zero. After the preprocesing process, **Steinberger and Jezek (2004)** approach is applied. The second method for the Ozsoy approach is to find main concepts and sub-concepts in the **Topic Method**. In this method, similar to the Cross method, the SVD calculation step is used to pre-process the sentence selection step. Then the step to find the main topics is operated. A concept matrix is created for this. Each concept has strength values. In this way, main concepts and sub-concepts are found.

2.4.2 Graph Theory Based Algorithms

Graph theory is a mathematical representation of the connections between objects. The theorem was first processed by mathematician Leonhard Euler in 1735. Graph theory consists of graphs and vertices. Nodes are entities, and the edges are links between nodes.

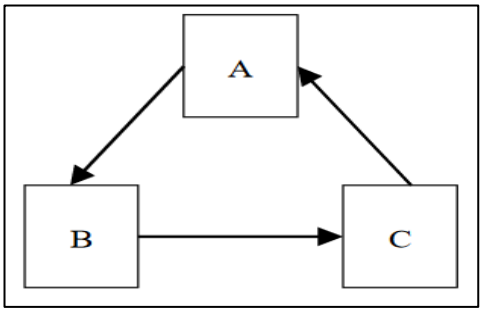
2.4.2.1 PageRank Algorithm

PageRank (PR) is a graph-based algorithm that Google uses to rank search results. The algorithm was published by the founders of Google, **Lawrence Page** and **Sergey Brin**. PageRank measures the number and quality of links to a page to estimate how important a page is. **(Page et al., 1998)** The following formula is used to calculate the PageRank value for page A.

PR(A) = (1-d) + d (PR(T1)/C(T1) + … + PR(Tn)/C(Tn)) **(2.3)**

Equation **(2.3)** shows PageRank of page A, PageRank of T1 pages linking to **PR (T1)** A page, **C (T1)**, number of links given to other pages from T1 page, and dumping parameter ranging from 0-1 d. PageRank is an iterative algorithm. At the first time the values of all pages are the same. PageRank values of the pages are determined by the result of iterations. For example, let's say we have 3 pages named A, B and C and d is 0.7.

Figure 2.1: PageRank example website referencing



Source: Dell Zhang, 2006, PageRank Examples

Assume that the pages are linked together as in the way. We see that page A is a reference from page C which shown as **Figure 2.1**. The C page does not have a reference to another page. The PR value would be 0 if there was no link between the pages. If you apply the PageRank formula **(2.3)** for these pages for A page.

PR (A) = (1 – d) × (1 / N) + d × (PR (C) / 1)

Similarly, page B also refers to page A. Page A has no other page reference. In this case, **C (Tn)** value specified in PR formula is 1. We can find the value of PR (B) as follows.

PR (B) = (1 – d) × (1 / N) + d × (PR (A) / 1)

The page C also takes a reference from page B as shown. The B page does not have a reference to another page. Apply the PR formula to find the PageRank value of the C page.

PR (C) = (1 – d) × (1 / N) + d × (PR (B) / 1)

When we put the given values in the formula **(2.3)**,

PR (A) = (1 – 0.7) x (1/3) + 0.7x(PR (C)/1) → PR (A) = 0.1 + 0.7 × PR (C)

PR (B) = (1 – 0.7) x (1/3) + 0.7x(PR (A)/1) → PR (B) = 0.1 + 0.7 × PR (A)

PR (C) = (1 – 0.7) x (1/3)+ 0.7x(PR (B)/1) → PR (C) = 0.1 + 0.7 × PR(B)

If we solve the equations, we calculate the PR values of the pages.

PR (A) = 1/3 = 0.33 PR (B) = 1/3 = 0.33 PR(C) = 1/3 = 0.33

2.4.2.2 TextRank Algorithm

The TextRank algorithm **(Mihalcea et al., 2004)** is based on the PageRank algorithm and is a graph based algorithm. As mentioned on PageRank, reference relations between pages play an important role in determining the importance of the page. The main idea is that the pages that are important are referenced from the pages that are important and the PageRank of a page determines the probability of a user visiting that page. We can use the PageRank algorithm in text summarization applications. The PageRank algorithm can be adapted to extract a text summary. This is primarily based on sentences rather than pages. Therefore, the TextRank algorithm contains sentences instead of pages. The similarity between the two sentences can be considered as a reference to another page in the PageRank algorithm. Similarity scores between sentences can be kept in the similarity matrix. A summary of a given text can be made by making a sort on these similarity values. If we examine the basic steps used in the TextRank algorithm.

Step 1: Pre-processing is performed on the text. Text is then divided into sentences.

Step 2: A language model is created from sentences. While creating the language model corresponding to the sentences, BOW, TF-IDF or GloVe can be used as one of the ready word vectors.

Step 3: Similarity values in sentence vectors are stored in the similarity matrix.

Step 4: Similarity matrix is converted to a graph. Graph in vertices (nodes) sentences, similarity scores are shown as edge. Similarity measure can be used as cosine similarity.

Step 5: The sentence is selected according to the similarity scores listed in the graph.

2.4.2.3 LexRank Algorithm

The LexRank algorithm **(Erkan et al., 2004)** is based on the PageRank algorithm like TextRank and also it is a graph based algorithm. The main idea is that the sentences recommend other similar sentences to the reader. The importance of sentences is also due to the importance of a sentence. In this way, a sentence should be similar to many other sentences to be selected as a summary. This summation approach is called **Centrality-based Sentence Salience**. The main difference between LexRank and TextRank algorithm is the weighting function, which assigns weight to the graph edges. TextRank accepts all weights as unit weight, as in PageRank, and calculates the sentence ranks accordingly. In other words, TextRank works like PageRank's one-to-one application. LexRank calculates the centrality of the sentence to assign weight. If we examine the basic steps used in the LexRank algorithm.

Step 1: Pre-processing is performed on the text. Text is then divided into sentences.

Step 2: The language model is created from sentences.

Step 3: Sentence vectors are converted to graph. In Graph, each sentence represents a node. The Edge represents the similarity relationships between sentences.

Step 4: The similarity between the sentences is found in the cosine formula modified by the TF-IDF formula.

Figure 2.2: LexRank idf-modified-cosine formula

$$\text{idf-modified-cosine}(x, y) = \frac{\sum_{w \in x,y} \text{tf}_{w,x}\text{tf}_{w,y}(\text{idf}_w)^2}{\sqrt{\sum_{x_i \in x}(\text{tf}_{x_i,x}\text{idf}_{x_i})^2} \times \sqrt{\sum_{y_i \in y}(\text{tf}_{y_i,y}\text{idf}_{y_i})^2}}$$

Source: LexRank: Graph-based LexicalCentrality as Salience in Text Summarization Erkan et al., 2004

The formula which is shown at **Figure 2.2** measures the distance between the x and y sentences. Similar sentences are closer to each other.

Step 5: A similarity matrix is created in which sentence scores are stored.

Step 6: According to similarity matrix values, a sort is created and sentence selection is made.

2.4.3 Luhn's Algorithm

It was one of the first studies in the field of text summarization in 1958 published by **Hans Peter Luhn**. The algorithm is also used in various fields such as credit card number checksum operations. The Luhn algorithm is based on the TF-IDF model. According to Luhn, some words are frequently used when the subject is detailed by the author. Frequent use of the word

indicates the importance of that word for writing. A word can be given more importance to each of these words if one word is found in a text. If we examine the basic steps used in the Luhn algorithm.

Step 1: In the first step, the keywords are determined. The minimum and most commonly used words are determined according to the frequency of use in the text. The most used words and the less used words are ignored.

Step 2: Weight is calculated for each sentence in the text. Weight values are the score values for sentences. To find the score, we need to calculate the value of window size first. Window size value in the sentence between the two important keywords are showing the distance value. The score value is calculation of square of important keyword count division by windows size value.

Step 3: The calculated score values for each sentence are sorted. Then the sentence selection is made.

2.5 ABSTRACTIVE METHODS

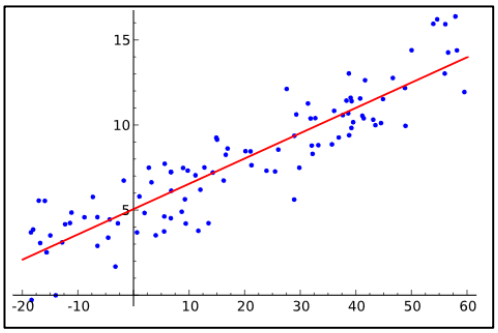
2.5.1 Deep Learning Prerequisities for Text Summarization Application

There are some issues that need to be examined before starting the text summary application. Deep learning practices need to be built on these foundations. In this section, I will show you the important issues that should be examined in order to do text summarization using deep learning techniques. Linear regression is the basis of regression analysis. As we progress by reducing estimation errors, we can say that it creates the starting level for deep learning.

2.5.1.1 Linear Regression

Linear regression is a type of analysis used to measure the relationship between two or more variables in statistics. On the data set, we aim to find the optimum line that passes through the data. We can estimate the value we want by this line which is shown as at the below **Figure 2.3**.

Figure 2.3: Linear regression line fitting



Source: (Wikipedia, 2019) [1]

As we know from mathematics, the equation of the line as follows.

Y = b0 + b1 * X. **(2.4)**

In the equation **(2.4) X** is the x-axis and y is the y-axis. For **b0** we can say the point where the line is the point y. b1 is the slope of the line. In literature, m can be shown in different sources. When we pass a line through the data like in the figure, it will be away from some points near some points. Our goal is to find the most optimal line that passes through the data, we need to find the difference between the real value and the estimated value.

Residual = y – ŷ **(2.5)**

In equation **(2.5) ŷ** indicates the estimated value. **y** is our real value. Residual is the difference between the real value and the estimated value. This calculation needs to be calculated separately for each point. In doing so, our goal is to find the most appropriate line by reducing the error. It's also called line fit. When calculating the residual values, some values fall under the line and get negative values according to the result of "**ŷ** – y". In order to find the total error, we need to take square of result. In this way, we are not experiencing loss due to negative values. Mathematically expressing our operations.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(yi - yi\hat{})^2 \quad \textbf{(2.6)}$$

The Mean Squared Error (MSE) equation **(2.6)** shows the sum of errors calculated for each point. Yi - yi ^ is the residual value for each data point, and the n parameter is the sample count,

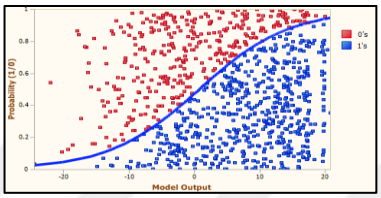
[1] Linear regression.2009. https://en.wikipedia.org/wiki/Linear_regression [Accessed date 03 May 2019]

ie the number of points in our data. By minimizing the values obtained using the MSE formula, we obtain the most appropriate line. In other words, our goal is to find the min (MSE) value.

2.5.1.2 Logistic Regression

Logistic regression is a binary classification algorithm and is simply a neural network. In this sense, we can say that the basis of deep learning. The linear regression aims to find the best straight line, while logistic regression aims to classify the data on the best S curve like **Figure 2.4**. Makes a probability estimate as output.

Figure 2.4: Logistic regression data classification

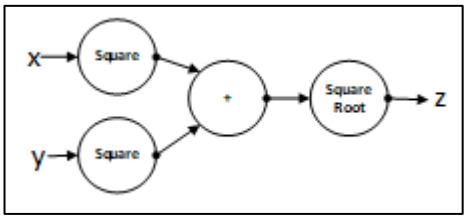


Source: (Veribilimcisi.com, 2019)[2]

In fact, the difference between a deep learning structure and a logistic regression structure lies in that the how much deep. The word deep is related to the use of more neurons in the network. The depth of an artificial neural network is a relative concept. In order to visualize the calculations of the artificial neural network network, diagrams called **computation graphs** are used. We can see one instance in **Figure 2.5.**

For example, if we want to show z = $\sqrt{x^2 + y^2}$ using computation graph.

Figure 2.5: computation graph



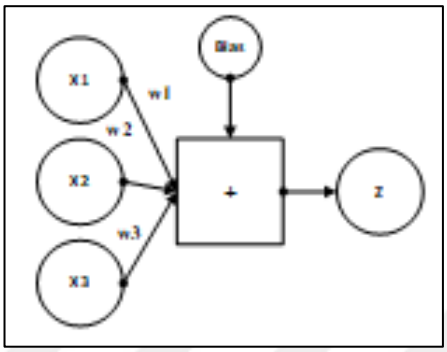
X and Y input, while the circle is to show the operations. Computation graph reduces the complexity of the process.

[2] Logistic regression data classification.20018. https://veribilimcisi.com/2017/07/18/lojistik-regresyon/ [Accessed date 03 May 2019]

If we define a neuron as a mathematical equation. To specify X input value, W weight, B bias and Z, we can write the following equation to indicate the output. **(Everything you need to know about Neural Networks - hackernoon.com)**

$$Z = X*W+B \quad (2.7)$$

Figure 2.6: Basic neural network

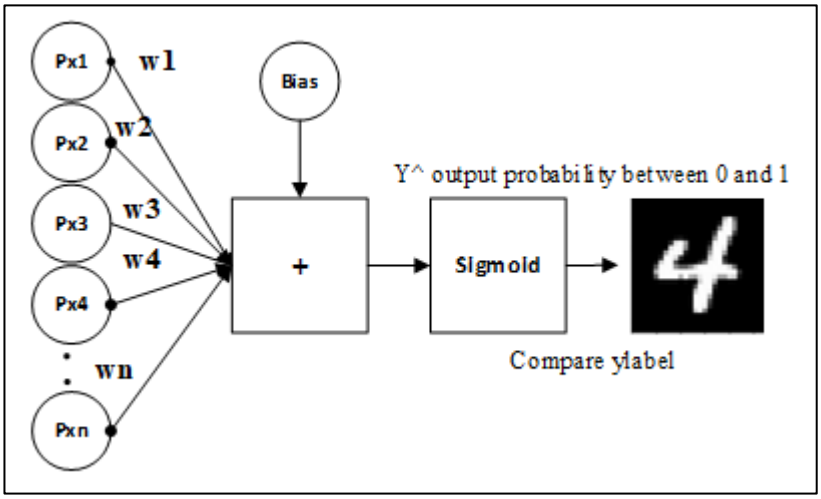


By using equation **(2.7)**, we describe a neuron, we can define a neural network consisting of 3 neurons and 1 layer above. In this case, our neural network equation is transformed into matrix multiplication as follows **(2.8)**.

$$Z = [w1, w2, w3] * \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} + b \quad (2.8)$$

We can think of weights as values that indicate the importance of connections. For example, if the weight of the x3 input is higher, the output will be more affected by the x3 input. If we back to the logistic regression, when we add sigmoid to the neural network structure consisting of 1 layer above, our output becomes a system that gives a statistical result between 0 and 1.

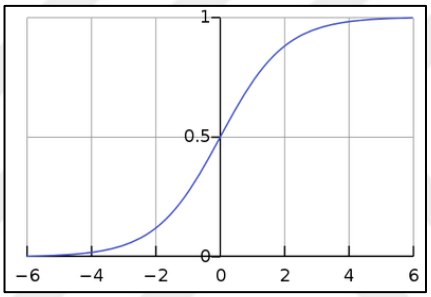
Figure 2.7: Logistic regression structure



Using computation graph, we can construct a logistic regression structure like above **Figure 2.7**. Our logistic regression model tries to learn weight and bias. In the example above, we try to estimate the label values of the numbers in the form of images in the **MNIST** dataset. The input is given the pixel values of the picture we know the value of the system. The aim is to try to understand what the picture is. We multiply each pixel of the image with its weight values. Then the input values multiplied by weight are collected and then the bias value is added. The result is given to the sigmoid function.

The **sigmoid** function equals the results to a numbers between 0 and 1. The sigmoid function is graphically as follows. One of the two reasons for using Sigmoid is that it can be differentiable and the other reason is producing probabilistic results.

Figure 2.8: Sigmoid activation function



Source: (Wikipedia, 2019)[3]

We are trying to predict the result by comparing the result with the threshold we have determined. For example, let's say that we have found 4 as a result of z. Assume that the value of 4 is 0.9 in the sigmoid function. In this case, we can say that the result is 90% probability 4. If we estimate the value, we need to verify by comparing the actual label.

2.5.1.2.1 *Forward Propagation*

To summarize the steps of logistic regression, we take the input of the pixel values of an image as input. We apply the $\mathbf{Z = [w1, w2, w3] * \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} + b}$ **(2.9)**

operation for each pixel input. We give the result of the process as a parameter to the sigmoid function, which returns a probabilistic result. We compare our actual value with the result. If we guessed correctly, our Loss value would be 0, but if we guessed wrong, our Loss value would be high. The entire process from the start to the Loss function is called **forward propagation.**

[3] *Sigmoid function.*2019. https://en.wikipedia.org/wiki/Sigmoid_function [Accessed date 03 May 2019]

2.5.1.2.2 *Loss Function*

We can define the loss function like at the below.

$$\mathbf{L(\hat{y}, y) = - (y \log \hat{y} + (1 - y) \log(1 - \hat{y}))} \quad \textbf{(2.10)}$$

The purpose of the loss function **(2.10)** is to minimize the error between the estimated value and the actual value. For instance, suppose that our actual value and the estimated value are 1.If we replace the values in the loss equation. We find L(1,1) = -(1log1 + (1-1) log(1- 1)) =-(0+0.log(0))=-(0+0) = 0 value. In this case, there is no loss value. Assuming we made a wrong estimate, L(0,1) = -(1log0 + (1- 1) log(1 - 0)) =- (infinity + 0) = infinity We find infinity result. In this case, we can understand that our loss value is quite high. We collect a loss value for each input and get a cost value. We understand that we need to update our weight and bias values by looking at our cost. We aim to reduce the cost value by updating the weight and bias value. If we summarize the forward propagation flow.

$$\mathbf{X, w, b \rightarrow Z = [w1, w2, w3] * } \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} \mathbf{+ b \rightarrow A = \sigma(z) \rightarrow L(y, \hat{y})}$$

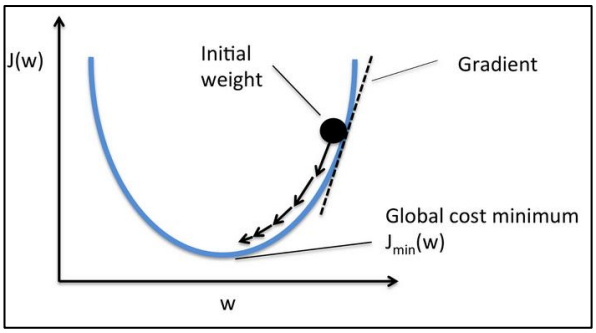
2.5.1.2.2 *Backward Propagation*

In order to decrease the cost value obtained in the cost function, we need to update the weight and bias values. We can define the cost function as follows **(2.11)**. We need to find w and b values to minimize cost.

$$\boldsymbol{J(w,b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}(i), y(i)) = - \frac{1}{m} \sum_{i=1}^{m} [(y(i) log(\hat{y}(i)) + (1 - y(i)) log(1 - \hat{y}(i))]} \quad \textbf{(2.11)}$$

This is called backward propagation. The method we will use in the backward propagation process is Gradient Descent. Gradient Descent is an optimization algorithm. We need to be able to give the optimum values to the parameters so that we can reduce the cost to the lowest value. We can do this with optimization algorithms. **Adam** and RMSprop are a few of these algorithms.

Figure 2.9: Gradient Descent



Source: (hackernoon.com, 2019)[4]

We see w values corresponding to J (w) cost values as above. The initial weight on the curve is far from the global cost minimum. We aim to reach the minimum cost point by optimizing W and b values. The gradient is called the slope that cuts the cost value vertically which shown at **Figure 2.9**. We can see that we reach the minimum point when the slope is zero or close to zero. Each time the backward and forward propagation process is repeated, the slope will be approaching a little bit more. The back propagation is trying to get a partial derivative operation from the value obtained from the loss function to the input. The derivative of a function according to a point gives the slope of the function. By updating the w and b values we find the minimum value of the cost function. In this way, we are able to make accurate estimates by training our model. The learning process is provided in this manner.

$$\boldsymbol{w = w - \alpha \frac{\partial j(w,b)}{\partial (w)}} \qquad \textbf{(2.12)}$$

Equation **(2.12)** tells us that we get the derivative of the cost function based on the w value. W represents the weight value, b represents the bias value. α learning rate, in other words, the size of how much it will take a step in each iteration. The learning rate parameter is a parameter that we must specify. According to the results we have to change the learning rate.

$$\boldsymbol{b = b - \alpha \frac{\partial j(w,b)}{\partial (b)}} \qquad \textbf{(2.13)}$$

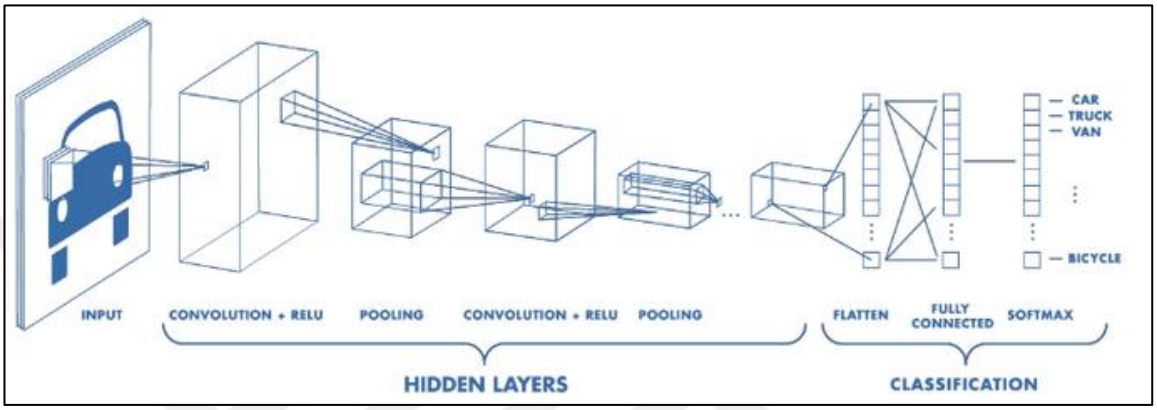
We make similar calculations in b (bias) for the equation **(2.13)**. In this case, we need to take the derivative of the cost function.

[4] *Gradient descent.*2018. https://hackernoon.com/gradient-descent-aynk-7cbe95a778da

2.5.1.3 Convolutional Neural Network (CNN)

Convolutional neural network is one of the two basic algorithms of deep learning and the other is recurrent neural network (RNN). The CNN algorithm is a classification algorithm like logistic regression. It is often used Image classification, object identification, image and captioning problems.

Figure 2.10: Architecture of CNN



Source: (mathworks.com, 2019)[5]

Convolutional neural network architecture consists of 3 main parts as seen above. There is a car picture as input. There are hidden layers for finding the attributes in the car image, and finally there is the classification section where the classification is performed.

2.5.1.3.1 *Convolution Operation*

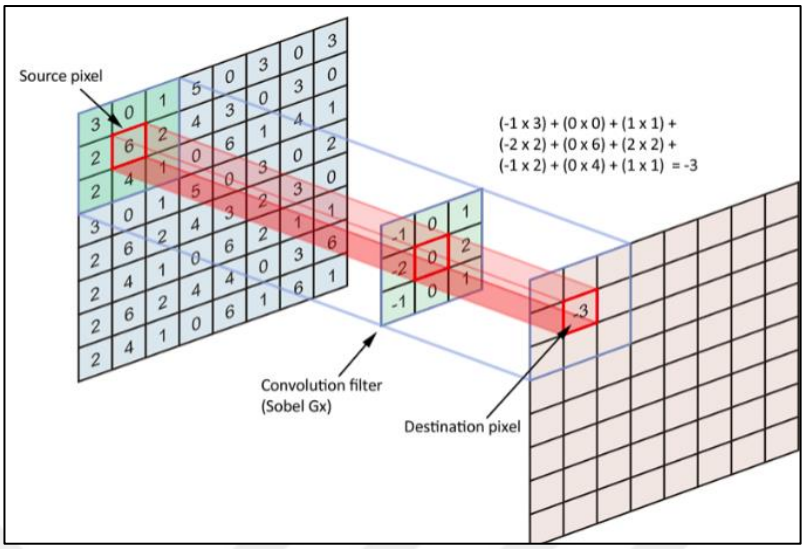
The Convolutional layer is equipped with filters that allow us to find features through the input. As is known, the input is actually a matrix. We need to use feature matrices to find feature in this matrix. Feature matrices are called kernel. If the input is a car, it allows us to find the car's headlights or wheels. **Convolution** is a mathematical function that expresses how a shape is altered by the other. Math is defined by the following formula **(2.14)**.

$$[f * g](t) \equiv \int_0^t f(\tau)\, g(t-\tau)\, d\tau, \qquad \textbf{(2.14)}$$

Convolution process is subject to 3 elements. Input image, convolution filter, ie feature detector and feature map.

[5] Architecture of cnn.2018. https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html [Accessed date 03 May 2019]

Figure 2.11: Convolution operation



Source: (freecodecamp.org, 2019)[6]

Convolution process as shown above **Figure 2.11** input on the image is made by shifting the convolutional filter. Overlapping numbers are multiplied. The numbers obtained as a result of multiplication are collected and transferred to the feature map matrix. This process reduces the size of the input image, causing data loss. The **same padding** method is used to prevent data loss. The same padding tries to add a frame of zero values around the input image. This prevents data loss. Feature map stores a feature, such as the car's headlamp, so we need numerous feauture maps to identify all features in the input image. After the Convolution process, an activation function relu is applied to the feature map. The reason for inserting the process into the relu function is to reset the negative values. If we examine the **Relu** activation function **(2.15)**,

$$f(x) = \begin{cases} 0 \ for \ x < 0 \\ x \ for \ x \geq 0 \end{cases} \qquad \textbf{(2.15)}$$

Result is zero for values less than zero as in the expression. For values equal to or greater than zero, the result is unchanged.

[6] *Dertat,A.*,Convolutional operation.2018. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2 [Accessed date 03 May 2019]

2.5.1.3.2 *Max Pooling*

Max pooling is a down sampling method applied to reduce the number of parameters. It is also used to solve the overfitting problem. A window is passed through the dimensions specified on the Feature map matrix. It is a process done by taking the max value within the window.

2.5.1.3.3 *Flattening*

The process of converting the resulting matrices after the convolution and pooling operations into n-line 1-column vectors is called flattening. These vectors will be the inputs of the artificial neural network.

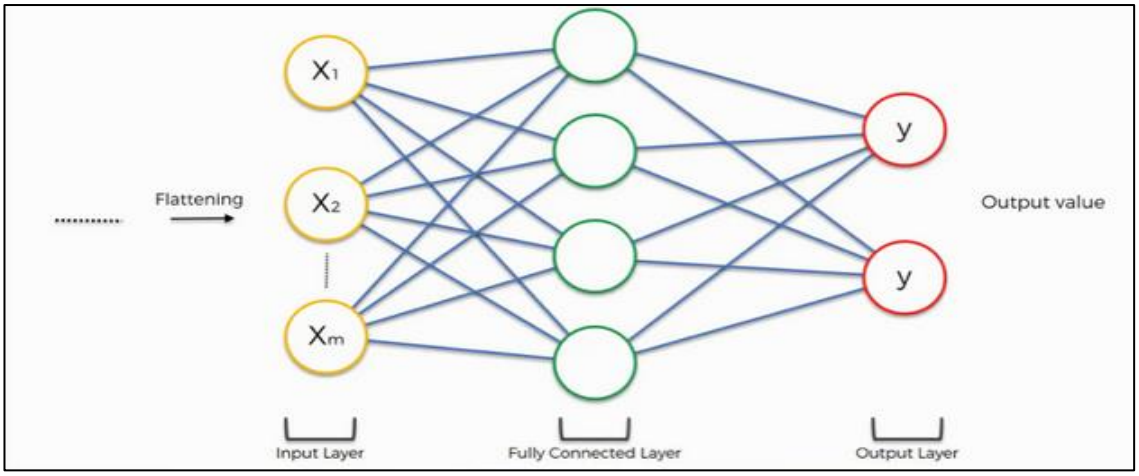
2.5.1.3.4 *Fully Connected Layer*

Fully connected layer is the part of artificial neural network. It is the layer where learning through artificial neural networks occurs. Therefore, as in logistic regression, forward propagation and backward propagation take place in this layer.

2.5.2 Recurrent Neural Network (RNN)

RNN is one of the two basic structures of deep learning along with CNN. RNN is used for time series. For example, voice processing, speech to text, text to speech, chatbot, machine translation and text summarization.

Figure 2.12: CNN Fully Connected Layer



Source: (superdatascience.com, 2019)[7]

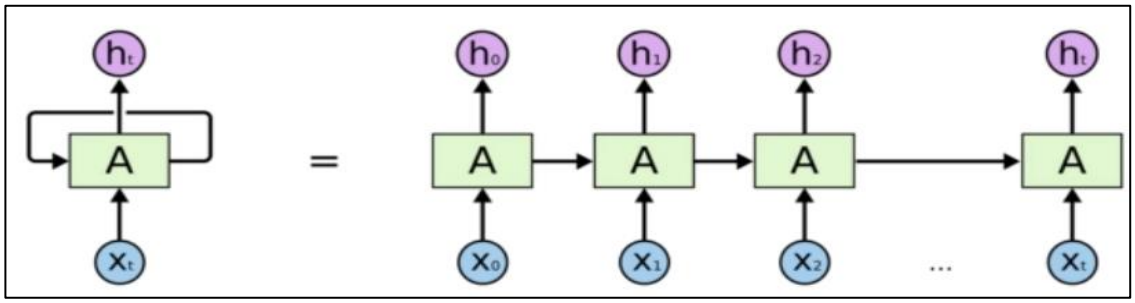
[7] *Cnn fully connected layer*.2018. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection [Accessed date 03 May 2019]

If we remember the CNN architecture which is shown **Figure 2.12**, the neural network is receiving a certain size of data. It processes this data in the neural network. An output is generated as a result of the process. For example, by taking picture data as input and returning the class of the image as output.The human brain isn't made up of billions of neuron cells that work in one direction only. From this perspective, the CNN architecture remains a fairly basic architecture compared to the human brain's data processing structure. Not only what we saw or heard at that moment in making a decision. It is also necessary to consider the reasons and consequences of the decisions taken in the past.

While CNN is very successful in class-based transactions, it doesn't seem to be successful enough in the data that we use when talking about successive times. The order of words in a sentence and the order of sentences in the text is very important. The ranking in the data in the CNN is not very important. Since the order in the data is ignored and there is no memory to remember the data in the past, the data is currently being processed. When an input is received again, the previous operand data does not matter and is forgotten.

In the RNN architecture, a memory is added to the artificial neural network to remember the results of the operation in the previously operated time steps. We can simplify the structure of the artificial neural network as shown below. A loop is added to this structure to form the memory structure. The loop in RNN provides RNN self-feeding. If we open the loop in Rnn, we see that it consists of time steps which is shown at the below **Figure 2.13**.

Figure 2.13: RNN Time Steps



Source: (Colah, 2019)[8]

X: Mars is the fourth planet from the Sun and the second smallest planet in the Solar System.

[8] Olah, C., 2015. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/ [accessed date 03 May 2019].

Assuming that we're dealing with the sentence above, each word in the sentence represents $\mathbf{X_{0,\ldots,X_t}}$. This series also includes punctuation. **x (t)** t input t, **h (t)** t shows the time of the hidden status. After the input **x (t)** is processed in the A neural network, the hidden state is updated and then an output is generated. If we examine the Rnn equation **(2.16)**,

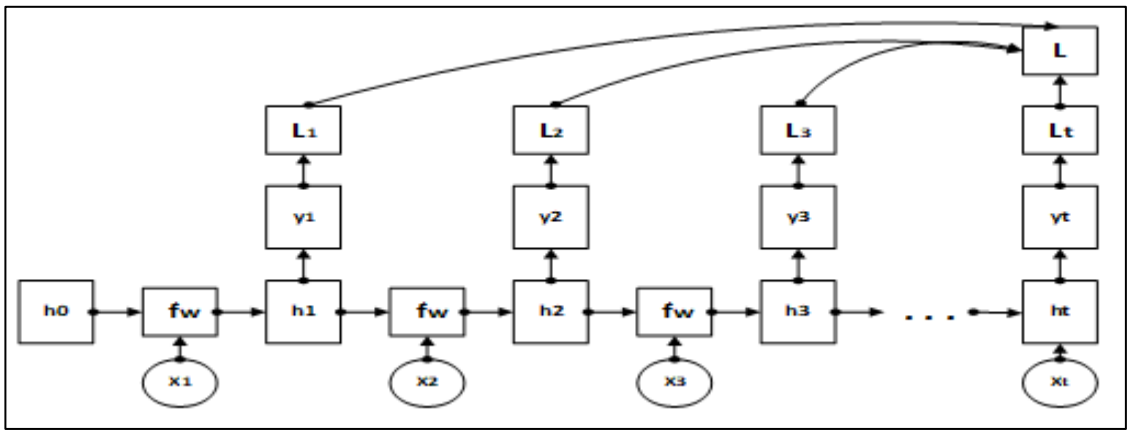
$$h(t) = f(h(t-1), x(t)) \qquad (2.16)$$

The new status h (t) is determined by the previous state and input h (t-1), which are the parameters of the function f. In other words, the function takes the input vector of the hidden state and t at the time of the previous one. A new hidden state and output will be generated when the RNN is executed. This process is repeated at each time step. In this way a memory is transmitted on the rnn.

2.5.2.1 Forward Propagation

Forward propagation process in Rnn is performed as follows. If we examine the **Figure 2.14**, the rnn function always takes a hidden state with an input x (t) in the step. In the initial condition, we can initialize the hidden state as the 0 vector. Hidden state and input are given to neural network system. The output is produced as a Y output and a current hidden state. The newly produced hidden state is again given to the neural network with the next X input. Output and hidden state occurs as output. This continues until the input sequence is finished.

Figure 2.14: RNN Forward Propagation



During forward propagation, the neural network always uses a different input and hidden state, but uses the same weight value. A Loss calculation is required for each output value generated. We need Loss values for the predicted y (t) values in our neural network. We refer to the loss values as L in the figure. In forward propagation, we calculate a Cost value by collecting the Loss values generated at each time step. The process of calculating the cost value is called forward

propagation. The loss function allows us to see how accurate the output the model is. After all input values are processed, the resulting Loss values are collected and we get a cost value.

2.5.2.2 Backward Propagation

We need a cost value for back propagation. If the estimated result y (t) is different from the actual value "y", our Cost value will be high. If the cost value is high, we understand that we need to update our W and B values. Our goal is to minimize the Cost value. The process of updating Weight and Bias values is called back propagation. We can show the loss function mathematically in the following way.

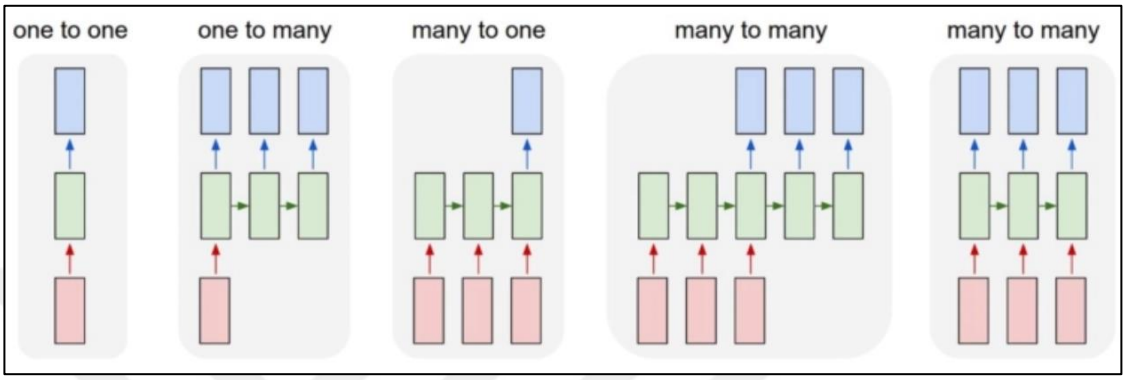
$$L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log (1 - \hat{y})) \quad (2.17)$$

In equation **(2.17)** y is the actual value ŷ is the value we guessed. In the first iterations, it is expected that the cost value will be high. Because the initial values are given intuitively. Depending on the W and Bias values, Loss values will be large. The back propagation is going backwards in time steps. We use optimization methods to optimize back propagation by reducing errors. Generally used methods are **Gradient Descent** and **Cross Entropy** methods. For Gradient Descent, we can say the method of reduction according to its derivative. Using the optimization algorithm, first calculate the L values for each time step. The L value is actually a counterweight to our Weight and bias values. Then y values and h values are calculated. As we move forward in forward propagation, our outputs are L y and h, and we have output values w and b in back propagation. In this way, the current W and b value again we can perform forward propagation. We can say that learning happens in back propagation part.

2.5.2.3 Different RNN Architectures

There are situations where the input sequence is not equal to the output sequence according to different needs and areas of use. Therefore, different types of RNN have emerged as shown **Figure 2.15**.

Figure 2.15: RNN Types



Source: (karpathy, 2019)[9]

One to one: In this type of rnn we have an input for the t times and we have an output for that t time. For this architecture we can call x (t) = y (t). We can give the picture classification as an example of this architecture. We get an image as the input and we return the class of that picture.

One to many: In this type of rnn, the input is an input of a single moment, while the output is produced for the moment and the next moments. We can give Image Captioning as an example of this architecture. We give input as image and we get the words that describe what happened in that picture as output.

Many to one: This architecture gives more than one input and we get a single output. The example of many to one architecture can be sentiment analysis. We give a text to the syntax analyzer. As a result, we expect this article to be positive or negative and expect an output to be produced. **Many to many:** In many to many architecture, the number of inputs and outputs can be multiple and different. As an example of this architecture we can give text summarization or machine translation. As a long text input and output as a short text expect.

Other than that, there are second many to many architecture. The difference is that after each input an output is generated. In the case of previous many to many, the inputs are first processed and then output is generated. For example, input may be a video, we may want to make a classification on every frame of the video. In this case, we can use this many to many styles.

[9] *Karpaty, A., Rnn Types.*2015. http://karpathy.github.io/2015/05/21/rnn-effectiveness/ [Accessed date 03 May 2019]

2.5.2.4 Creating a Language Model with Rnn

*"I want to **sell** my phone.", "My **cell** phone has been broken."*

When we examine the above two sentences, we see that the meanings of the words "sell" and "cell" are the same. In English, these words are called **homophones**. We can find the possibility of using the two words according to the frequency of the text.

*P(I want to **sell** my phone.) = $6.7x10^{-7}$, P(My **cell** phone has been broken.)= $1.7x10^{-13}$*

As above, the word "sell" may be more likely to be found in the text. In addition, the word "sell" after the word comes with the word "sell" we need to look at the possibility. Accordingly, when we try to find the probability of the whole sentence P (sentence) = (y1, y2,….,yty) we need to specify the output y for each word. We need to add extra **<EOS>** tokens when tokenize the words. EOS Token is a sign that the sentence is over. We also consider punctuation marks in sentences as a token.

In American Indian mythology the lynx is considered a 'keeper of secrets'. <EOS>

The word **lynx** in the sentence is a difficult word to find in the dictionary. Some special words may not be available due to the few words in the dictionary. In this case, words not found in the dictionary are referred to as **Unknown (UNK)** words. Detailed information for this dictionary is in the **5.2.7 Word embeddings** section.

If we try to create an RNN model for the sentence above. If the zero vector x (1) is the input 0 because we do not know the first word of our sentence at the first moment is not a value before we get zero. With the value h (1), we estimate y (1). We're looking at the word pool we created to predict the first word. Any of these words can be considered as a candidate for y (1) output. At Y (1) output, the probability is calculated using softmax. We use the value of y (1) as the input in the next time step. Using the word "In", we look at the words that can come after the word "In". After the word " In American Indian ", the probability of the word 'mythology' is calculated. For example, after the first two words in our sentence, we multiply their probability to find out what the third word is.

$P(y(1),y(2),y(3)) = P(y(1))* P(y(1)| y(2)* P(y(1)| y(2)|y(3))$

Source: (Andrew Ng – Language model and sequence generation coursera course notes)[10]

By this way the series continues until the EOS token is got, so the sentence is finished. From this situation, we understand that the word order has an effect on the previous word, the next word,

[10] Ng, A., Language model and sequence generation coursera course notes, 2019 https://www.coursera.org/lecture/nlp-sequence-models/language-model-and-sequence-generation-gw1Xw [Accessed date 03 May 2019]

all the words before a word itself. RNN has learned a word in each time step. Loss values that occur at all moments are collected using the Loss function. As previously mentioned, we compare the actual value with the estimated value to find the actual loss values.

Since this input takes a word in each step, this whole model is called a word-based language model. One disadvantage of this model is that if we do not have a word dictionary in the input, this word is written as Unknown and it causes distortions in probability calculations in subsequent words. The character-level language model can be used, although it is not very efficient to solve this problem.

Using the character-level language model, we can find words that are not in the dictionary. This stands out as the most important advantage. The most important disadvantage is that the number of transactions and resource problems are increased since it is character based.

2.5.2.5 Exploding and Vanishing Gradient Problems

As in CNN in the RNN architecture, learning takes place in the back-propagation process. In this process, the value from the Cost function is compared with the actual value. As a result, an optimization algorithm, such as Gradient Descent, is used to update the back-propagation weight and bias values. Two Gradient problems are encountered in the back propagation stage at RNN. In order to be able to update W and bias values in RNN cells, we multiply the W matrix in each cell when we want to derive according to Loss. The W matrix has the same value in all cells. When we multiply a value with itself, there are two cases. If this value is greater than one, it reaches very large values in a short time and this is called **Exploding Gradient**. If it is smaller than one, it will continue shrinking and it will come to the point of disappearance. This situation is called **Vanishing Gradient**. For larger networks, multiplication processing will be much more. In this case, the possibility of encountering these problems will increase.

A method such as **Gradient Clipping** can be used to solve the problem of Exploding Gradient. In this method, a **threshold** is determined and the Gradient is prevented from passing this value. This method is often used in simple RNN. There is a simple RNN solution for Vanishing Gradient. If this problem is encountered, a different type of RNN is required. RNN types such as **LSTM** and **GRU** should be used to eliminate these problems.

2.5.3 GRU (Gated Recurrent Units)

GRU is one of the RNN architectures developed for solving Vanishing and Exploding Gradient problems. **(Cho, et al., 2014)**

GRU solves the vanishing gradient problem with structures called update and reset gate which are shown at **Figure 2.16**. **Update gate z_t** which is shown at equation **(2.18),** applied sigmoid 0 is the previous memory, if 1 is a new value with the current memory is updated.

$$z_t = \sigma(W_z\,[h_{t-1}, x_t]) \qquad (2.18)$$

Thus, vanishing gradient problem is prevented. In addition, the plurality or singularity of the words remaining in the past, location or contact information is also transmitted. The **reset gate r_t (2.19)** uses it to calculate the candidate information to be added to the memory. He predicts how relevant the memory candidate is to the previous memory. He decides how much of the history is forgotten.

$$r_t = \sigma(W_r\,[h_{t-1}, x_t]) \qquad (2.19)$$

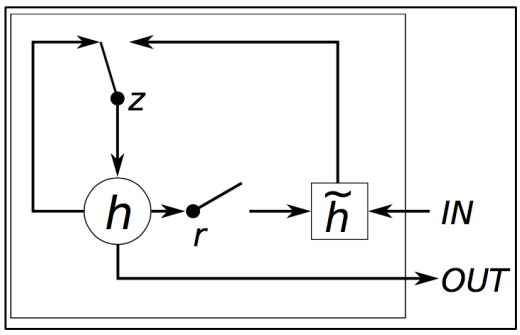
If we examine how the reset and update gate affects output, a new memory content is available to use the reset gate to store the relevant information in the past. This process is calculated as follows.

$$\hat{h} = \tanh(W\,[r_t * h_{t-1}, x_t]) \qquad (2.20)$$

This determines what will be deleted from previous time steps. As a last step, it is necessary to calculate equation **(2.21)** the h_t value that holds the information for the current rnn unit and transmits the information it holds to the whole network. The update gate is needed to do this.

$$h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h} \qquad (2.21)$$

Figure 2.16: Gated Recurrent Unit



Source: (Chung, Junyoung, et al., 2014)

2.5.4 LSTM (Long Short Term Memory)

LSTM is a type of RNN with long-term memory. **(Hochreiter, et al., 1997)** It is a generalized version of GRU. The advantage of LSTM over basic RNN is that it has both long and short term memory. If we try to predict the last word in the sentence **"the clouds are in the *sky*,"** This is easy to predict. RNN learns to use past words. If the sentence like that **"I grew up in France… I speak fluent *French*." Source: (colah, 2019)**

In other words, he says that he grew up in France at the beginning of the sentence. He says he can speak French fluently at the end of the text**.** RNN is insufficient in that kind of sentences. In theory, RNN 'France' and 'French' need to be able to establish the relationship, but as the sentence lengths RNN can not learn this relationship. LSTM overcomes this problem with its short and long-term memory. In LSTM, there are gates affecting the output as well as GRU, forget and exit gates are used instead of the reset passage in GRU which are shown as **Figure 2.17**.

$$f_t = \sigma(W_f\,[h_{t-1}, x_t] + b_f) \qquad (2.22)$$

In the **forget gate f_t (2.22),** LSTM decides which one of the information must be forgotten. Sigmoid function makes the decision-making process. If we examine the equation in time t-1 hidden state and t input is input. While most of the information close to the sigmoid result 0 cannot passes through the gate, the calculated information that is close to one is transmitted.

$$i_t = \sigma(W_i\,[h_{t-1}, x_t] + b_i) \qquad (2.23)$$

$$\hat{C}_t = \tanh(W_c\,[h_{t-1}] + b_c) \qquad (2.24)$$

Input gate i_t (2.23) decides what information to update. The Tanh layer creates a vector of candidate values to be added to the cell state. Then the cell state **(2.24)** is added by combining two equations. Thus, it is determined which new information should be added to the cell state. After completing the above 3 neural network operations, the old cell state's Ct needs to be updated.

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \qquad (2.25)$$

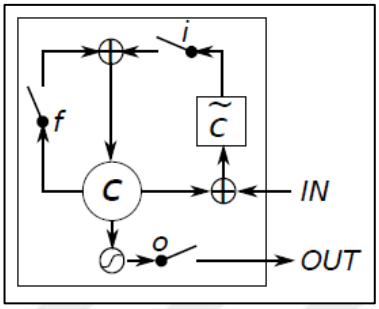
t-1 time on the cell state (C_{t-1}) is done in the forget gate process. In this way, the information that should be forgotten from the cell state. **Input gate i_t (2.25)** and cell state candidate values state to be added to the cell state information was required to be added. New $\hat{C}_t$ is being created by summing operation. Finally, the output gate **o_t (2.26)** is decided by what should be the output.

$$o_t = \sigma(W_o\,[h_{t-1}, x_t] + b_0) \qquad (2.26)$$

$$h_t = o_t * \tanh(C_t) \qquad (2.27)$$

The output gate determines which information is given as $\mathbf{o_t}$ **(2.26)** output. Then the cell state tanh function is passed **(2.27)**. Tanh is an activation function that produces values in the range of -1 and 1. When we process the output value with the output gate value, we give the parts of the cell state as output.

Figure 2.17: Long Short Term Memory



Source: (Chung, Junyoung, et al. 2014)

2.5.5 Bidirectional RNN

Suppose we sort the words in the sentence by type. We can label the words in the sentence with labels such as people, place name, company name and date.

For instance;

"General relativity is an exciting theory about the physics of space and time.**"**

We can't label the word General in any way. If our sentence was in the following way.

"General Lee and the Confederate Army lost the great battle."

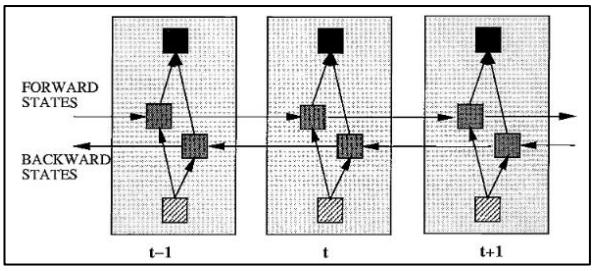
We can label the word general as human

"**General** Motors is a car producing company." Here we can label General Motors as company name.

While LSTM and GRU are guessing the words in the sentence, they will act as a single word for the word "General" and will not find that this word is a human name or a company name according to the meaning of the sentence. If these sentences were shown to a person, person could easily understand it. Therefore we can see the whole sentence together at the same time. RNN can see past information, but cannot see future information.

To solve this problem, the bidirectional RNN structure is used. Firstly, we perform the same calculations in RNN architecture in the same way as the bidirectional RNN structure. This RNN architecture can also be in GRU or LSTM.

Figure 2.18: Bidirectional RNN



Source: (Schuster, Paliwal, et al. 1997)

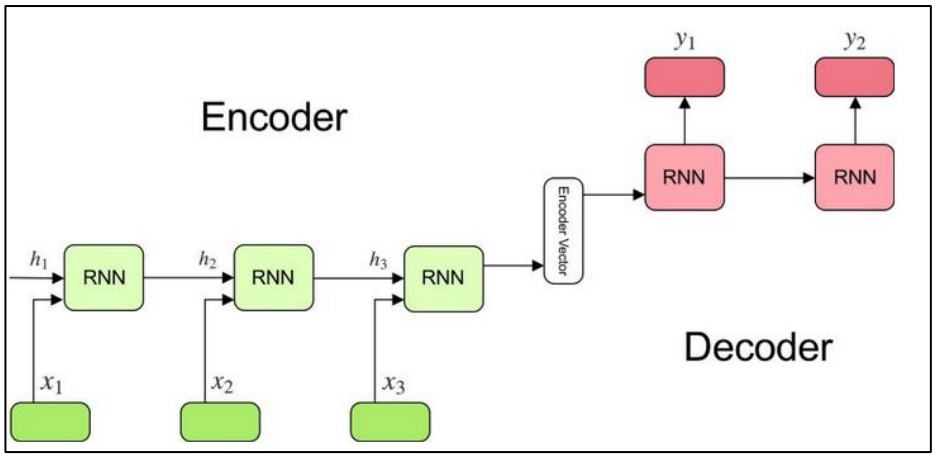
In order to be able to see the future information and predict the exits of BRNN, forward and backward operations must be completed as shown at **Figure 2.18** and all hidden state values must be calculated. The process that takes place at this stage is independent of the forward and backward propagation process. Forward states calculated in the forward direction depend on their previous forward state values, while the backward state value calculated in the reverse direction is again dependent on the previous backward state value. In other words, when calculating an output, we benefit from both past and future data. BRNN is very successful in situations where there is a certain text in our hands, but we won't be successful if we don't have any future knowledge. We should consider this when developing the model.

2.5.6 Sequence to Sequence (Seq2Seq) Models

Let's consider that we want to make applications like Text summarization, machine translation or chatbot using RNN. Considering what is common in these applications, we see that a text is entered into the system and another text is returned as output. The size of the input and output text can be the same or they can have different word counts. For example, in Text summarization, we expect to give a relatively large text and a smaller text that is relevant to this text. In this case, the input to the system is a sequence, output is a sequence. Such models can be called seq2seq models.

The Seq2Seq architecture consists of two interconnected RNN systems as shown at **Figure 2.19**. The first RNN system gets input, it is called the encoder. The second RNN system produces output, this system is called Decoder. It does not have to be input text received. Audio, picture and video can be given to the system such as input.

Figure 2.19: Encoder-Decoder Seq2Seq model



Source: (towardsdatascience.com, 2019)[11]

2.5.6.1 Encoder

The encoder working like a standard RNN but does not produce a prediction as output. It only keeps the last state. A summary vector is generated from the sequence given as input. In this vector, we can say that the time information is an illustration of the input sequence which is the input sequence for the vector. The purpose of the encoder is to create a thought vector that takes sequence information and represents a smaller size and sequence. For a encoder vector or context vector, we can say that it conveys a smaller representation of the input sequence to the decoder. It helps decoder to make more accurate predictions.

2.5.6.2 Decoder

If we talk about how the decoder works, the decoder is an RNN system like encoder. But the RNNs in this system have their own weight and bias values. It should have the same size as the vector it receives from the encoder. Therefore the status information from the encoder will create the initial status of the decoder. Any word is given as a starter to the decoder. It should be noted that this word is a unique word that is not included in the sequence. In general, the <SOS> token is used as a starting token but this is an optional choice. Initial output is estimated using the starting vector and the context vector from the encoder. In this way, the values are transferred and a word is estimated at each step.

[11] Kostadinov, S,. Seq2seq architecture.2019. https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346 [Accessed date 03 May 2019]

The Seq2Seq is an end-to-end model although it consists of two parts RNN. In other words, after calculating the loss values resulting from the forward propagation, the weight and bias values generated in the encoder should be updated by starting from the decoder for back propagation.

2.5.7 Word Embeddings

One of the main problems we will face when working on Natural Language Processing (NLP) is to symbolize words or characters in a way that the machine can understand. We need to be able to articulate our words or characters in order for the machines to work on the text we give. When we express our machine text numerically, it becomes able to perform all kinds of operations on the text.

First, we need to create a vocabulary dictionary in which the words are expressed numerically. The size of the dictionary depends on us. For example, in Turkish or English, we can create a dictionary of 1000 words that are the most common in daily conversations. We need to define one hot vector for each word in our dictionary. Vectors representing one element 1 and other elements 0 to represent the word are called one hot vector.

The vector length increases in proportion to the number of words. It is necessary to create 1000 long vectors for 1000 words. In applications requiring high capacity vocabulary such as text summarization, this number should be much more. Also, if we express the words in terms of only one hot vectors, the meaning between words will be lost. In other words, the relationship between words with one hot vectors is not stored.

In order to understand a word, we need to understand its connection with other words. For example, the word 'cell' is associated with the word 'biology' or the word 'cell' is associated with the word 'phone'. But the word 'cell' is low relation on the word 'road'. We cannot keep these meaning relationships between words in one of our hot vectors that we have created in our dictionary, and we cannot determine a relationship level through vectors.

We need to use vectors to represent the words in order to indicate the relationship between the words. But these vectors should be better than one hot vector notation. In other words, the numbers in the vector will not only be 0 or 1.

The closeness of the numbers between the two words will hold the relation. The vectors of two similar words will be close to each other. We find the connection of two words to each other by multiplying two vectors with each other. The process results in a scalar number. We can say that the larger the number, the more relevant the two words. **Word Embedding** is the way that words

are shown in such a way as to keep their meaning in relation to each other. The word embedding matrix has a word vector for each word.

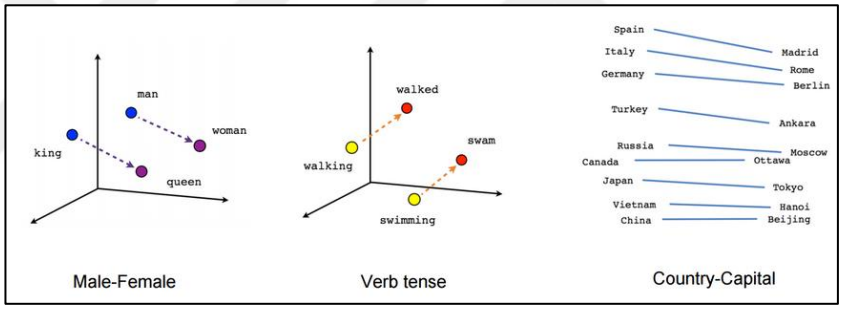
Figure 2.20: Word representation

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

Source: (Andrew Ng – Word representation coursera course notes, 2019)

When we examine the **Figure 2.20**, we can establish a link even if we do not know the connection between the King → Queen and the Man → Woman duo. Similarly, we can see Apple → Orange connection. The vectors of King → Queen and Man →Woman are close to each other. Therefore, we can say that the man → woman is approximately proportional to the King - Queen subtraction process.

Figure 2.21: Relationships between words



Source: (tensorflow.org, 2019)[12]

Techniques such as **Cosine Similarity** or **Euclidian Distance** can be used to find similarity between Man→ Woman and King→ Queen. Similarly, human names, such as Istanbul → New York, can be linked between city or place names which are shown at **Figure 2.21**. Vocabulary gives you the number of words found in our dictionary, and dimension refers to the number of features embed. The features change according to the corpus we process, so it can be used in feautures like gender, age, food, or feautures like place and name. These methods are the **Word2Vec** and **Glove** methods. One of these two methods is used when creating word vectors.

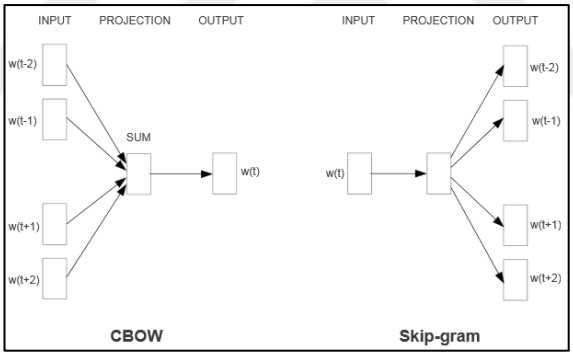
[12] 2015. https://www.tensorflow.org/tutorials/representation/word2vec [Accessed date 03 May 2019]

2.5.7.1 Word2Vec

Words around a word can be useful in finding the meaning of that word. We can use this feature when creating word vectors. For example, the word “cell” will be used to create a vector based on which words are used. In order to accurately predict the meaning of the word “cell”, the word2vec model needs to be trained on many sentences which passes 'cell' word in text. If we can guess the words around a word, we'll find the meaning of the target word. Similarly, when people hear a word for the first time, they try to guess the word they do not know the meaning of the whole sentence. Web2Vec is one of the methods used for this purpose. Web2Vec is one of the unsupervised learning methods found by Google. The words at the edge in Word2Vec represent the middle word.

The Word2Vec method has two different algorithms. These algorithms are **Skip-gram** and **Continuous bag of words (CBOW)** which are shown at **Figure 2.22**. Skip-Gram algorithm predicts words on the edge of the middle words. **(Mikalov, et al, 2013)**

Figure 2.22: CBOW and Skip-Gram Architectures



Source: (Mikalov, et al., 2013)

SkipGram structure takes the middle word as input and output gives the words around the middle word. The word can be predicted by the word distance which is called window size. If the window size is two, two words are taken from the right and left of the middle word. Because many similar sentences are seen together during the training and so word pairs improves the probability of words being together in same sentences.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(wt+j|wt) \qquad \textbf{(2.28)}$$

In the above equation **(2.28) T** shows the position of the word, **c** is window size, **w** is words. Our goal is to maximize the probability of finding words on the edge according to the middle word. The basic skip-gram formula that uses softmax is to optimize the vocabulary vectors and reduce the cost.

$$p(w0|wI) = \frac{\exp(v'woTvwI)}{\sum_{w=1}^{W} \exp(v'wTvwI)} \quad \textbf{(2.29)}$$

In the equation **(2.29)**, the index of the word in the **Wo** window, the center of the **WI** center word, the vector of the **V** middle word, **V'** shows the vector of the word in the window. Probability distribution is calculated with this formula. In the formula, Softmax is used for probability distribution. There is a problem when using Softmax. The denominator in the formula contains the whole word vector. When the formula is calculated for each word on the Corpus, considering the size of the corpus. When we try to calculate, the transaction load is increasing. The method used to speed up the calculation is **Negative Sampling**. **(Mikolov et al., 2013)**

For example, we are moving on the text to find the words in which the word "Cell" is linked. The word cell will not actually be side by side in most words. In our loss function, we use sigmoid instead of softmax to make our skipgram model interested in learning only high quality vector representations. Instead of taking all the words in the vocabulary and executing the total process, we process fewer words. Randomly selected words are found with unigram distribution. More frequent words are more likely to be selected. **Continuous bag of words (CBOW)** algorithm predicts the middle word from the words on the edge. The CBOW algorithm works in a very similar way to SkipGram. CBOW is trying to predict the middle word according to the size of the window given the words in the sentence in the form of the opposite of SkipGram. These operations are repeated from the beginning of the given text to the end with the words.

2.5.7.2 Glove (Global Vectors for Word Representation)

Glove model was published by Standford University in 2014. In Word2vec we update the word vector when the words "deep learning" side by side on the corpus, and we increase the likelihood that these two words will coexist probability. In Glove, the words "deep learning" are counted together in the corpus. Then we optimize the word vectors according to this counting.

$$J = \sum_{i,j=1}^{V} f(xij)(wi\hat{w}j + bi + bj - logxij)^2 \quad \textbf{(2.30)}$$

The loss function for Glove is as above **(2.30) (Pennington et al., 2014)**. We can use Gradient Descent to minimize cost. When we examine the function, J shows all parameter values. Like word2Vec, there are wi and wj vectors. The wi and wj vectors show any line and column of the matrix in which we get word counts. We create a matrix using word counting methods on all corpus. X refers to the generated matrix. f is an upper limit for the words that are frequently encountered in the corpus. In this way, stopword is counted up to this limit. These words are not counted if they exceed the limit. In addition, if the value of Xij is 0, the problem is log 0 undefined and the problem is solved with f is equal to 0.

3. EXPERIMENTAL STUDIES

3.1 DATASETS

Within the scope of the thesis, a total of 3 dataset was used. The first one is a large dataset **"WikiHow: A Large Scale Text Summarization Dataset" (Koupaee et al., 2018)**. WikiHow is more developed for abstractive summarizing methods. Besides, it contains more than 230000 articles and summary. The second dataset, **Amazon Fine Food Reviews**, has more than 560000 products and food reviews **(McAuley et al. 2012)**. The third dataset is **News Summary**, with more than 4500 news articles, news articles and news headlines. Dataset available on **Kaggle.com**

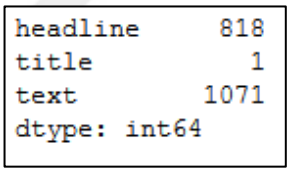
Source: (news summary dataset, kaggle.com)

3.2 DATA EXPLORATION

3.2.1 WikiHow

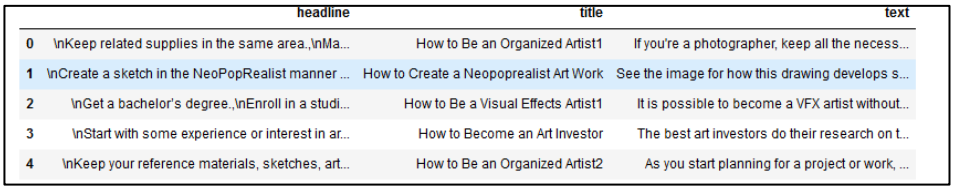
The WikiHow dataset consists of 215365 rows and 3 columns named headline, title, text. When null is checked in dataset, the following number of null values was encountered shown as **Figure 3.1**

Figure 3.1: WikiHow null checking



Null values are replaced with empty string values.

Figure 3.2: WikiHow dataset view



	headline	title	text
0	\nKeep related supplies in the same area.,\nMa...	How to Be an Organized Artist1	If you're a photographer, keep all the necess...
1	\nCreate a sketch in the NeoPopRealist manner ...	How to Create a Neopoprealist Art Work	See the image for how this drawing develops s...
2	\nGet a bachelor's degree.,\nEnroll in a studi...	How to Be a Visual Effects Artist1	It is possible to become a VFX artist without...
3	\nStart with some experience or interest in ar...	How to Become an Art Investor	The best art investors do their research on t...
4	\nKeep your reference materials, sketches, art...	How to Be an Organized Artist2	As you start planning for a project or work, ...

General view of the WikiHow dataset As above **Figure 3.2**, the headline column contains a summary of the topic, the title of the topic in the title and the details in the text. Title section was removed because it will not be used.

3.2.2 Amazon Fine Food Reviews

The Amazon Fine Food Reviews dataset consists of 568454 rows and 10 columns. The general view of the dataset is as follows **Figure 3.3**.

Figure 3.3: Amazon Fine Food Reviews dataset view

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

Only the Summary and Text columns were used in the columns. Other columns were not included in the study. When a null check was made in Dataset as above **Figure 3.4**, a null value was encountered in the following numbers on a column basis.

Figure 3.4: Amazon Fine Food Reviews null checking

```
Id                      0
ProductId               0
UserId                  0
ProfileName            16
HelpfulnessNumerator    0
HelpfulnessDenominator  0
Score                   0
Time                    0
Summary                27
Text                    0
dtype: int64
```

These null values have been replaced with empty string characters.

3.2.2 News Summary

News Summary dataset consists of 98401 lines, two columns called headlines and text. The overall image of the dataset is as follows **Figure 3.5**.

Figure 3.5: News Summary dataset view

	headlines	text
0	upGrad learner switches to career in ML & Al w...	Saurav Kant, an alumnus of upGrad and IIIT-B's...
1	Delhi techie wins free food from Swiggy for on...	Kunal Shah's credit card bill payment platform...
2	New Zealand end Rohit Sharma-led India's 12-ma...	New Zealand defeated India by 8 wickets in the...
3	Aegon life iTerm insurance plan helps customer...	With Aegon Life iTerm Insurance plan, customer...
4	Have known Hirani for yrs, what if MeToo claim...	Speaking about the sexual harassment allegatio...

When null check is performed in dataset, no null values are found.

3.3 DATA PREPROCESSING

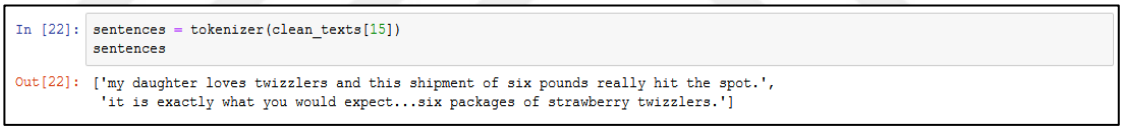
3.3.1 Data Cleaning

Data preprocessing was performed before creating the language model in WikiHow, Amazon Fine Food Reviews and News Summary. For this purpose a function named clean_text was created. The clean_text function takes contradictions parameters as boolean and parameter text as string. The text given as a parameter is converted to lower case by **lower()** method. If the **Contradictions** parameter is set to true, the expressions such as “don’t, won’t” in english are converted to expressions like “do not, will not”. This enables more consistent language modeling. In addition to this, regular expression was applied to text to remove special characters from text. In this way, special characters, the url format, letters, non-latin characters in the text was cleared.

3.3.2 Language Modelling

As explained in Section 2.2 Language Modeling, firstly texts were converted to tokens with the **tokenizer** function. The python **ntlk** library was used for the Tokenization process.

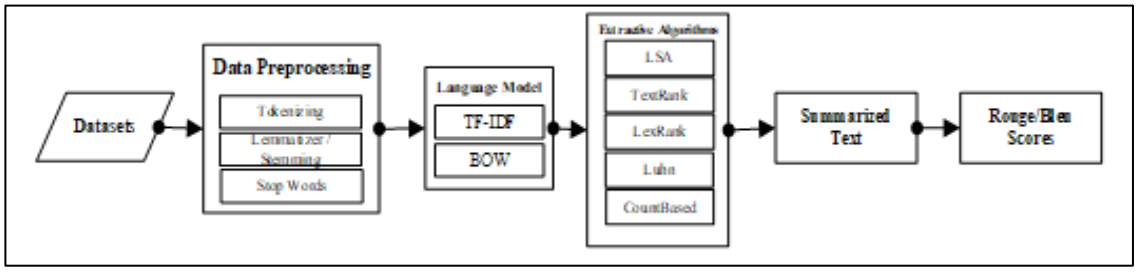
Figure 3.6: Tokenization process



Lemmatizer and stemmer functions in different forms using words, words were reduced to the roots. The **WordNetLemmatizer** function of the nltk library was used for the Lemmatizer function. For Stemmer, stemming was done with the **PorterStemmer** function. Frequently used words in other words Stop Words such as “a, about,the” omited from the text for more consistent models can be developed. The stopwords function was used for this operation. Text, including the words a, about, such as stop word, often more consistent models can be developed because of the text is exited. The stopwords function was used for this operation.

3.4 EXTRACTIVE METHODS

Figure 3.7: Extractive Methods process flow



Extractive text summarization methods were developed in the flow shown above **Figure 3.7**. In summary, data preprocessing process was applied for all dataset. Then the text was converted to tf-idf or bow matrices. In other words, the language model was created and the algorithm was implemented. 6 different text summarization techniques were applied to the language models. The abstract produced by the original text and summary pairs was first evaluated by eye and then rouge / bleu summary evaluation techniques.

3.4.1 Latent Semantic Analysis (LSA)

Lsa is an algorithm that makes classification on text according to the concepts in a text. We can also use this property in text summarization problems. In the first step, SVD algorithm is applied to input matrix created by Tf-idf or bow method. In this way, the input matrix A, U, S, V is divided into separate matrices. In the next step, the choice between the sentences of the most important concept is provided. **LsaSummarizer** function has been developed to perform these steps. LsaSummarizer function takes 2 parameters named text and summary. The data to be applied to Text LSA, and the summary to the actual summary information in dataset will be used to make a comparison with the LSA summary as a result of the summary transaction. For the SVD to be applied on the language model, **randomized_svd** library of the nltk library was used. The randomized_svd function takes the tfidf matrix as the parameter and returns the U, S, V matrices. In order to obtain Salience scores, we use the values corresponding to the **s** and **v** matrices for each sentence. This is mathematically expressed as the following,

$$\mathbf{SS} = \sqrt{\sum_{i=1}^{k} \boldsymbol{SiVi}^{t}} \qquad \mathbf{(3.1)}$$

Equation **(3.1)** SS expresses saliance score corresponding to each sentence. **(Sarkar, 2016)** We multiply the values found in the S matrix by the values in the **vt** matrix for each sentence with the number of K sentences. After the score values are taken, the index information of the highest score value is assigned to a variable. In the dataset, the corresponding sentence for this index value is printed as a summary output.

3.4.2 TextRank

TextRank is a summarization algorithm based on reference relationships between sentences. A similarity matrix is created after tokenizing and language modeling. In the next step, the **similarity matrix** is converted to a graph. Graph in vertices (nodes) sentences, similarity scores are shown as edge. For the transformation of the graph, the from_scipy_sparse_matrix function in the python **networkx** library was used. Similarity scores were calculated using the **pagerank** function in the networkx library. When the score values are listed, the index information of the first order point to the sentence in the sentence series. Thus, a sentence is selected. The selected sentence is printed on the screen.

3.4.3 LexRank

The LexRank algorithm works with the same logic as TextRank. The sentence given by other sentences is of great importance to others. The difference between the sentences is found with the modified **cosine similarity** formula. The similarity value found is considered to be the sentence weight. The formula used is different from TextRank. **Sumy** library was used for the text summarization application with LexRank.

3.4.4 Luhn Algorithm

Luhn algorithm is used to determine the sentence value of the words used in text is based on the frequency. For this reason, firstly we have to find frequently used keywords in the sentence. The **get_keywords** function was developed for this operation. The Get_keywords function takes a parameter with the sentence name. A word list is created by dividing the received text words. When the word coincides with the same word in the list, the word score is increased by 1. Thus, the word and word score is kept in a list. The most frequently used words are removed from the list of words. To get the sentence scores, you need to calculate the window size. Window value is obtained by the distance between the two important keywords. The **calculate_sentence_weight** function was created for this operation. The score values found are listed in order to print the summary with the highest value.

3.4.5 Count Based Method

The Count Based text summarization technique is very similar to the luhn algorithm. In the given text, a list is created by counting frequently passed words. A weight calculation is made for each word by dividing the frequency of the most common word in the text by the frequency values of other words. After that, the points given for the words in the sentence are summed on sentence basis and each sentence is given a score. Finally, the highest score is printed as a summary.

3.5 ABSTRACTIVE METHODS

Abstractive text summarization techniques are techniques based on the production of sentences by guessing the word after the first word given after training on a given dataset. Developed seq2seq modeling for sentence production. The **Seq2Seq** architecture consists of Encoder and Decoder structures. After the input sequence is given to the encoder, the Encoder is creating a Context Vector. The Context Vector is given to the Decoder as initial state. In the training phase we give the Decoder the actual summary data in our dataset. Then the Decoder produce the output as a summary.

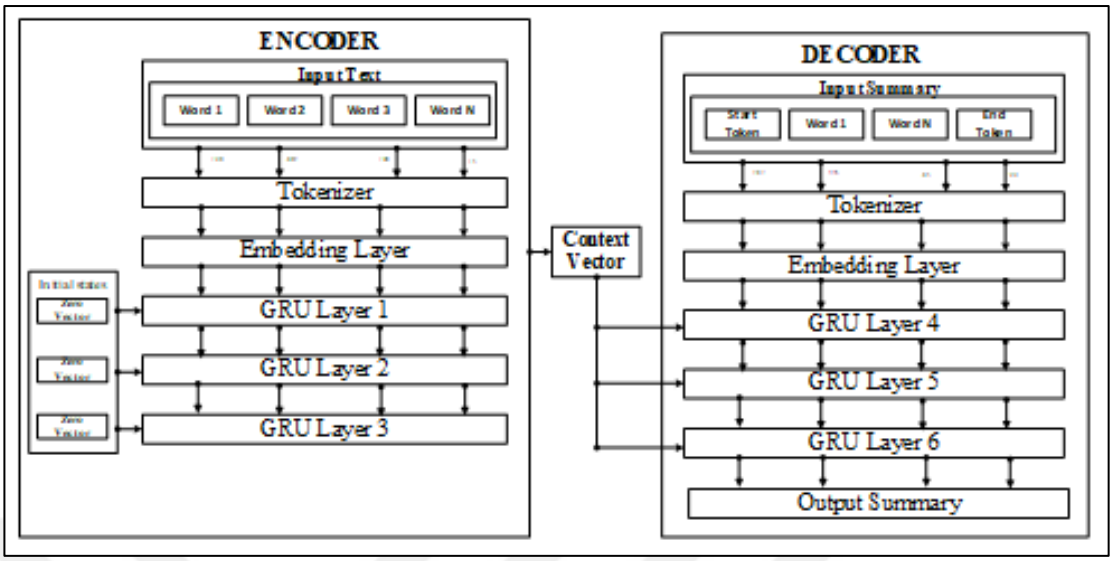
3.5.1 Seq2Seq Architecture

If we examine the structure of the encoder which is shown **Figure 3.8**, we give the text to be extracted as input. We need to apply tokenizing to the given text in order for the text to be processed. The words are converted to numeric values after the token operation. The **Embedding Layer** has a word vector for each word passed through token. **Glove** was used for the Embedding Matrix. In the Embedding Matrix we give the corresponding vectors the input to the RNN units. GRU structure was used as RNN. In addition, RNN structures consisting of 3 layers were used in **Seq2Seq**. We give 0 as the initial value for the RNN units. The first time step starts in these conditions. **RNN** units update their hidden states as they always encounter data in the step. Creating a Context vector when the calculation is finished in the encoder.

In the decoder's structure as shown **Figure 3.8**, the Context vector is retrieved from the Encoder, similar to the Encoder. Input is given as a start token when starting text and ending token when ending the sentence. Due to the start and end token, our model must start the sentence in the same way learns how to finish the sentences.

We set the start and end tokens. In making this selection, we have to select from the words that will not be included in the embedding matrix of these tokens. Training is not carried out properly if this is not done and can cause errors.

Figure 3.8: Seq2Seq Architecture



There is a vector within the embedding matrix of the initial token that should be given to the Decoder, as in other words. We give the Context vector from the Encoder to the RNN layers. Our model produces a word according to this information. Our goal is to draw the summary of our model by choosing the right words. The encoder works the same way in the training and testing stages, but the Decoder is given the actual summary text in the test phase. As the output, the decoder produces the correct word. That's how it's trained. In the test phase, the model is only initial token and the context vector from the encoder. Expectation is to be able to predict the correct words by giving the word again as input in the progressive time steps.

3.5.2 Tokenizing

Tokenization process was done by defining Tokenizer object in **Keras** library. If we do not use the **num_words** parameter in the token function, all words in the dataset are used. Using the function **fit_on_texts** in the Tokenizer class, we load our text into the tokenizer. With the **texts_to_sequences** function, a number is generated for each word we give to the tokenizer. Since the inputs given to the Rnn units have to be the same size, we need to set the sentences of different lengths to the same length. We determine the length and the standard deviation of the token list we have. Sentences may be shorter or longer than the length we set. If the sentences are shorter than the specified length, we add zero to the same size. This process is called **padding**. If we add padding to the beginning of the sentence, if we add padding to the end of the prepadding is called postpadding. Prepadding applied sentences shown at the **Figure 3.9**. Prepadding for the encoder is applying postpadding to the Decoder. When there is a sentence that exceeds the input size, we cut the excess. This process is called truncating. If the words are cut from the beginning of the

sentence, it is called **posttruncating** if the words are cut at the end of **pretruncating**. This can cause data loss. The first part of the sentence is more important than the last part. In order to prevent the first words in the sentence from being cut, we reverse the sentence. Another advantage of this process is that the Encoder is able to project the Context vector more successfully as it sees the beginning of the sentence. In the Decoder, producing a sentence starts from the first word, so you can do more consistently. With the **Reversed** function, we are able to reverse the sentences. Padding process is done with pad_sequences function. **Pad_sequences** function to determine the maximum token length of our tokens, padding process or prepadding whether the process of prepending or post truncating process as a parameter is pretruncating or posttruncating. The return value from pad_sequences function is assigned to the **tokens_padded** property. Below is a sample of prepadding applied sentence.

Figure 3.9: Prepadding applied sentence

```
array([    0,     0,     0,  9824,   107,   129,  2373,     8,  2358,
         448,   323, 19610,  2331,  1607,  1170,    16,  2111,  2444,
          22,  3987,   239,     3,     2,  5626,    61,  1028,   482,
         871,  2709,  5037, 19610,     6,  1977,     1,  1461,   238,
           5,    72,    93,   982,    16,  1535,    22,  2486,  1887,
       13858,     3,    12,   781,  1606,     6,  2358,  1894,     4,
        1977,  9996, 20457, 11371,     6, 12581,     5,  8221,    24,
        7092, 27470], dtype=int32)
```

Tokens_to_word and **tokens_to_string** functions are written to understand that the given text is properly tokenizing. Token is taken as a parameter in the **tokens_to_word** function. Token retrieves the word from the **index_to_word** dictionary and returns it. The Tokens_to_string function creates a sentence from the given token. It finds the word that corresponds to the word received as a parameter (except for tokens with a value of 0), and then creates a list of words. The elements in the list are combined and a sentence is returned as a string. We use the text_to_tokens function to test the model after texting the model. The function is converting the given text into tokens.

The **text_to_tokens** function takes the tokenizer object as a parameter, the text padding type, the reverse parameter of the sentence, and the boolean value as the reverse parameter. The imported text is converted to tokens with the text_to_sequences function. Tokens can convert the numpy array upside down in the reverse parameter. The flip method is used for inversion of the numpy library. In the flip method, with the axis parameter 1, we declare that we want the rows to be reversed. If the corresponding parameter was set to 0, the process would be performed. According to the selection of the reverse parameter are throwing truncating post or pre. We then apply the padding process to the tokens using the pad_sequences function.

3.5.3 Encoder - Decoder Input and Output

If we remember the Seq2Seq text summarization architecture, the encoder will give the text to be summarized as input. The encoder generates a context vector consisting of states and this vector is transferred to the decoder. Context vector is the initial state for the decoder. The decoder takes the start token with the context vector, which uses two inputs to try to predict the next word. As we give the word in the actual summary text which should be in the decoder stage, we are looking at the actual values and updated to the end token if they are wrong. In this case, the input and output given to the decoder is the same. The only difference between input and output is that the input output is a shifted state. This means that the text is output as the output without the tokens. By using the tokens_to_string function of the console, we can control the input and output text by converting the tokens to a string.

If we evaluate the inputs and outputs given to the decoder via sentences, we give the context vector generated by the encoder to the decoder during the training and the ssstarttoken, which is the starting token. The decoder produces a word according to the initial conditions. Our expectation is to produce the word Bosch, or use close words like company, brand. After that, the bosch word is given to the system as input. Our expectation is to produce the word 'makes' as output. In this way, the ending token eeendtoken to continue. At each time step a loss value will be calculated and optimized after being compared with the actual value.

3.5.4 Word Embedding

GloVe was used for word vectors in text summarization. **Glove** word vectors, trained by Stanford University, contains 400000 words and the corresponding vectors. Instead of using all the 400000 words in Glove, we get the vectors corresponding to the words in our vocabulary from glove. There are 98137 text and 43661 text abstracts in our vocabulary. We do not have words in our vocabulary, so we create the embedding matrix with random values. If there is a word in our vocabulary in the glove vectors, we transfer the corresponding vector in glove to my own embedding matrix. Thus, the majority of the embedding matrix consists of glove vectors previously prone. The size of our embedding matrix must match the size of the Glove word vectors. So we were given 100 as embedding you. Word2vec specifies the name of the empty dictionary we created. We need to read the glove vectors from the txt file. Glove file name **glove.6B.100d.txt** file name on the 6B 6 billion token is tilted over 100d is the size of the word vectors. We are reading the file using Python file reading methods. The file contains a word and

the corresponding vectors on each line. To transfer the Glove content to the word2vec dictionary, we are looping each line to get the word and the vectors' values.

word2vec [word] = vec as the key to the dictionary by specifying the words as the value of the word we are assigning the vectors. As a result, we get a dictionary of 400000 elements. Then we need to create an embedding matrix with random values. Our aim here is to transfer the glove vectors corresponding to the words in the vocabulary into my own embedding matrix. In order to generate random values, we use the random function of the numpy library. The values we assign randomly will be between 1 and 1. This section may vary by choice. We determine the size of the embedding matrix based on the number of words and the size of the vectors. We create a loop to match the words obtained from the glove with the words in the word book. If a word in the vocabulary matches a word in the word2vec glove dictionary, the vector associated with random numbers in the Embedding matrix will be replaced by the corresponding vectors in the word2vec dictionary. If the word in our vocabulary is not between the glove vectors, random numbers will remain. We can see the size of the generated embedding matrix with the shape property.

3.5.5 Encoder

We start by creating the model with encoder. For the encoder, we define the input layer as following **Figure 3.10**.

Figure 3.10: Encoder input

```
encoder_input = Input(shape=(None,),name='encoder_input')
```

The input object is taking the shape and name parameters. **Shape** determines the size of the input. Because it is given as 'none', input can be different sizes. The **name** parameter is used to specify the name of the input. The encoder input generated values are tokened. Then we define the embedding layer for the encoder using the Embedding object in Keras as following **Figure 3.11**.

Figure 3.11: Encoder embedding

```
encoder_embedding = Embedding(input_dim=num_encoder_words,
                              output_dim=embedding_size,
                              weights = [embedding_matrix],
                              trainable=True,
                              name='encoder_embedding')
```

Embedding object is taking parameters as input_dim, output_dim, weights, trainable and name. The total number of words to be given to the input_dim encoder is output to the output_dim parameter. We set Embedding size to 100. We assign the embedding matrix that we created to the

Weights parameter. In this way we are loading our embedding matrix to keras. By giving the Trainable parameter true, we are declaring that the model can be trained. False can also be given to this parameter because we use Glove's trained word vectors. It is useful to assign True to this parameter because random values are used for words that cannot be found in glove in the embedding matrix. The Name parameter allows us to name the embedding layer.

GRU was used as the RNN type for the encoder. Before creating the encoder to the GRU layer, we set a **state_size** to **256**. We use the state_size parameter in all GRU layers. Owing to this parameter, all GRU layers will produce 256-dimensional output. We use **CuDNNGRU** instead of standard GRU units. The training time is shorter because the CuDNNGRU is trained on the GPU instead of the CPU. It consists of 3-layer GRU layers for the encoder.

Figure 3.12: Encoder CuDNNGRU layers

```
encoder_gru1 = CuDNNGRU(state_size,name="encoder_gru1",return_sequences=True)
encoder_gru2 = CuDNNGRU(state_size,name="encoder_gru2",return_sequences=True)
encoder_gru3 = CuDNNGRU(state_size,name="encoder_gru3",return_sequences=False)
```

We can give a name to gru layer with **name** parameter. The return_sequences parameter allows the processed sequences to be transferred to the next GRU. The last GRU unit return_sequences is defined as False. Since we want the last encoder layer to produce a single output, we defined the **return_sequences** parameter False as above **Figure 3.12**. The encoder thus creates the Context vector. GRU layers need to be connected to each other in order to work.

The **Connect_encoder** function allows you to connect the GRU layers and the embedding layer defined for the encoder. The linking of the layers starts with the assignment of the input layer to the layer variable. After the encoder_input is assigned, we pass the layer variable to the encoder_embedding object and replicate it to the layer variable. Similarly, we assign 3 GRU layers to the layer variable. Finally we return the assigned output encoder output. When we call the function, we get the context vector that we will give to the decoder.

3.5.6 Decoder

As mentioned before, the decoder takes the context vector from the encoder last layer and the actual summary tokens after the initial word token. First we need to define these inputs for the decoder as **Figure 3.13**.

Figure 3.13: Decoder input and initial state

```
decoder_initial_state = Input(shape = (state_size,),name='decoder_initial_state')
decoder_input = Input(shape=(None,),name='decoder_input')
```

The encoder generates a context vector in the size of the state_size. The input would take the Decoder's initial states so it's state_size-sized. Similar to the encoder input, we define the input in the decoder. The length of the vector that we set 'None' and we declare that any length vector is accepted as the input. Similar to the embedding layer for the encoder, we create an embedding layer in the decoder as **Figure 3.14**.

Figure 3.14: Decoder embedding

```
decoder_embedding = Embedding(input_dim = num_decoder_words,
                              output_dim = embedding_size,
                              name = 'decoder_embedding')
```

In the input_dim parameter, we give the number of words to be given to the decoder. We give the variable output_dim to the embedding_size variable. We defined Embedding_size as 100. Since we use Glove vectors, the vector length should be the same as Glove, so we set it to 100. We also name the decoder embedding layer with the name parameter.

Figure 3.15: Decoder CuDNNGRU layers

```
decoder_gru1 = CuDNNGRU(state_size, name='decoder_gru1',return_sequences=True)
decoder_gru2 = CuDNNGRU(state_size, name='decoder_gru2',return_sequences=True)
decoder_gru3 = CuDNNGRU(state_size, name='decoder_gru3',return_sequences=True)
```

Similar to the encoder GRU layers, we also define decoder GRU layers as **Figure 3.15**. state_size was set to 256. We return the return_sequences parameter to True so that the values returned from the previous decoder layer can be transferred to the next decoder layers. Unlike the encoder, we define the return_sequences parameter as True in the last decoder layer. False value was given because we expect the context vector to be returned in the encoder layer. Because we expect the Decoder to produce a sentence as a result of its operation, the return_sequences on the last layer are set to True. Decoder 256 is produced from the last layer layer. To convert the generated vectors into words, dense layer needs to be created as **Figure 3.16**. It converts the vectors from a layer gru layer into vectors from one hot.

Figure 3.16: Decoder dense layer

```
decoder_dense = Dense(num_decoder_words,
                      activation='linear',
                      name='decoder_output')
```

We give the number of words given to the decoder as a parameter to the dense layer. Linear activation was used as activation function. The linear activation function is used to transmit the decoder output in the same way. Different activation functions, such as Softmax, can be used

instead of the dense layer linear activation function. As in the Encoder, we create a function called **connect_decoder** to connect the layers together. The function takes a parameter named initial_state. The Initial_state parameter represents the context vector created in the encoder. We pass this parameter to the decoder layer layers. The process of linking the layers is similar to that of the encoder. First we connect the decoder input and then the embedding layer. We then integrate the layers of layers with other layers. We send the initial_states parameter as a parameter to the decoder gru layer. Finally, we define all layers. We call the generated function and pass it to the decoder_output variable.

3.5.7 Model Training

Encoder and decoder layers are defined. But to ensure that the model can be trained end-to-end, it is necessary to create a model that connects the encoder and decoder layers.

Figure 3.17: Model train process

```
model_train = Model(inputs=[encoder_input,decoder_input],outputs=[decoder_output])
```

Using the model object, we connect the encoder and decoder layers together as **Figure 3.17**. The model object takes the input of the encoder and decoder inputs as input. In the Outputs parameter, we give the output of the decoder layers. Owing to this model, the encoder and decoder can be trained end to end.We connect models as **Figure 3.18**.In addition to the model used in the training, we use two models to be used in the testing phase.

Figure 3.18: Connections of models

```
model_encoder = Model(inputs=[encoder_input],outputs=[encoder_output])
model_decoder = Model(inputs=[decoder_input,decoder_initial_state],outputs=[decoder_output])
decoder_output = connect_decoder(initial_state = decoder_initial_state)
```

The encoder model is taking encoder_input, which we have previously defined as input. The output is encoder_output. During the test, we give text to the encoder. The encoder will return the context vector as output. Similarly, we create the decoder model that we will use in the test. The decoder model uses the decoder_input in the list as input and the context vector from the encoder as the decoder_initial_state parameter. We expect the Decoder to produce a summary of the text given to the encoder as a output. We need to create a loss calculation function before we can test the model we have created. Tensorflow's **sparse_softmax_cross_entropy_with_logits** function was used as loss calculation method as **Figure 3.19**. The function takes two parameters, the first parameter is y_true actual values, the second parameter is y_pred if the predicted values.

Figure 3.19: Sparse cross entropy

```
def sparse_cross_entropy(y_true,y_pred):
    loss = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y_true,logits=y_pred)
    loss_mean = tf.reduce_mean(loss)
    return loss_mean
```

The parameters received are given to sparse_softmax_cross_entropy_with_logits. The function returns a loss value as much as the batch length that we specified during the training. In other words, if we train our model in batches of length 128, the input values of the given input are 128. We find the average value of the loss values so that we can calculate a single loss value. Then, we return the average loss value. **RMSprop** gives better results in applications using RNN. **Adam Optimizer** can be used as an alternative to RMSprop.We compile our model using Keras **compile** function. As we determined before, the training process is done on model_train as following **Figure 3.20**.

Figure 3.20: Model compile

```
model_train.compile(optimizer=optimizer,
                    loss=sparse_cross_entropy,
                    target_tensors=[decoder_target])
```

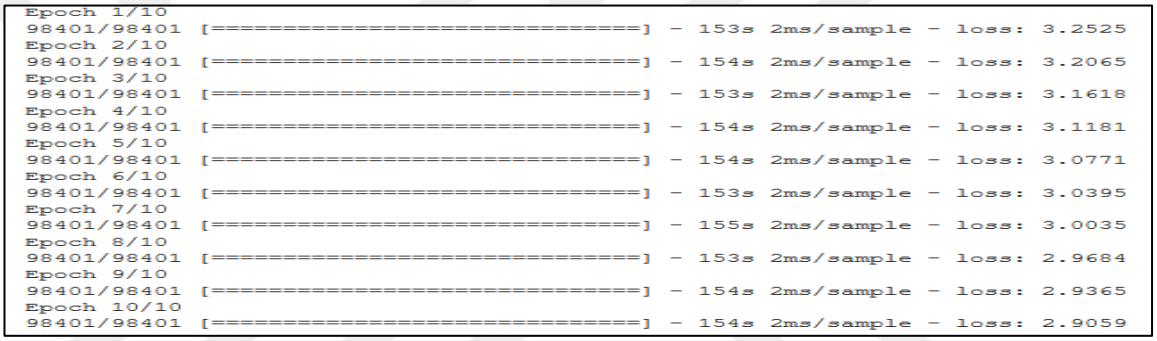
We assign the Optimizer parameter as RMSprop as we previously defined. Sparse_cross_entropy was used as loss function. We refer to the function loss parameter we define. In the target_tensors parameter, we assign an int32 type tensorflow placeholder named decoder_target. In this way, we make the model ready for training. Since the model training time is very long, we may want to save the model. In this case we can use the **ModelCheckpoint** object to store the weight and bias of the model on our disk. Using these values will save time to continue training the model.The fit function is used for model training. We prepare the parameters that we will give to the fit function as **Figure 3.21**.

Figure 3.21: Model fitting

```
x_data = {'encoder_input': encoder_input_data,'decoder_input': decoder_input_data}
y_data = {'decoder_output': decoder_output_data}
model_train.fit(x=x_data,
                y=y_data,
                batch_size=256,
                epochs=20,
                callbacks=[checkpoint])
model_train.save('model_more.h5')
```

x_data or encoder_inputunu and decoder_input. We are introducing the x_data variable that we define to the x parameter in the fit function and the y_data variable that we define in the y parameter. We set a batch_size based on computer power. Batch_size is a parameter that determines how many sets of data are trained in groups. Batch_size is set to 128. The Epoch parameter reports how many iterations will be trained. We're reporting that we want to train the model 10 times by giving Epoch 10. Callbacks parameter is the checkpoint. In this way, we will tell you where to save the trained model. Each time the model is trained, a checkpoint is generated, and if we report the weight values recorded on the checkpoint instead of starting from the start of the bending time, the training continues through the recorded model weight values.Under the conditions we set, training is taking place as **Figure 3.22**.

Figure 3.22: Training taking place



3.5.8 Model Test

After the model is trained, the summarize function is used to test. **Summarize** function extracts the desired text. Text is previously tokenizing. The tokenizer object's **text_to_tokens** function is used for this operation. As before the model is educated, we use the given text to invert and padding. For the model_encoder model, when the predict function is called with a tokenizing parameter, predictions are generated for the encoder and as a result the encoder model outputs a context vector. We keep the Context vector in the **initial_state** variable. We set max_tokens to prevent the decoder from entering the infinite loop.

Figure 3.23: Model prediction

```
initial_state = model_encoder.predict(input_tokens)
max_tokens = tokenizer_dest.max_tokens
decoder_input_data = np.zeros(shape=(1, max_tokens), dtype=np.int)
```

We assign the number of words we give to the decoder to the maximum token number. When the decoder reaches this number, it will no longer produce words. We are creating an empty array to give the input to the Decoder. We need to create a loop to make word production. The loop will

return until the end token is reached and max_token. As a result of each cycle we will get a word and translate the word to text. The loop will assign the starting token to the empty array named decoder_input_data that we created in the first step. As is known, two inputs are given to the decoder, the first is context vector in the name of initial_state, and the second is the array decoder_input_data. The decoder_input_data array contains only the initial token of the array in the first loop. We assign two variables that we have prepared for Decoder as x_data and input them to the decoder. Decoder model predicts the next word using the predict function. In the cycle, we give the decoder the words obtained in the previous steps in each step. Context vector remains constant. As the cycle continues, a new word is estimated and added to the list. We return the returned values from the token to text and add them to the string and return it as output.

4. RESULTS

In this section, It is examined the summarized results of **extractive** and **abstractive** text summarization algorithms on **WikiHow, Amazon Fine Food Reviews** and **New Summary** dataset that were introduced in section **3.1 Dataset**.

4.1 ROUGE AND BLEU SUMMARY EVALUATION METRICS

4.1.1 Rouge Metric

Rouge (Recall-Oriented Understudy for Gisting Evaluation) are the metrics used to evaluate automatic text summarization and machine translation software in natural language processing. The following is calculated using the formula.

Precision = $\frac{number\ of\ overlapping\ words}{total\ words\ in\ system\ summary}$

Rouge-N measures unigram, bigram, trigram or higher n-gram overlap measurements.

Rouge 1 word-based measurement.

Rouge 2 makes measurements on the basis of bigram words couples.

Rouge L measures longest matching sequence of words.

Let us say we have human summary like at the below.

Human summary: "I am going to school today."

If we assume that machine summary of above sentence as following.

Machine summary: "I am going to school."

Rouge 1 score is calculated as $\frac{5}{6}$ = 0.83

Rouge 2 score is calculated as the following.

Human summary bigrams:"I am","am going","going to","to school"," school today".

Machine summary bigrams:"I am","am going","going to","to school".

We count bigram word couples and calculated by same formula $\frac{4}{5}$ = 0.8

Rouge L is calculated by same formula, in this case **"I am going to school** today.**"** matches as longest matching sequence. $\frac{5}{6}$ = 0.83

4.1.2 Bleu Metric

BLEU (Bilingual evaluation understudy) are the metrics used to evaluate the quality of machine translation or automatic text summarization software. The **sentence_bleu** function of the nltk library was used to calculate the Bleu scores.

4.2 EXTRACTIVE METHODS RESULTS

4.2.1 WikiHow Results

Figures which are shown at the below shows results of the WikiHow dataset.

Figure 4.1: WikiHow LSA result

```
LsaSummarizer(clean_texts[100],clean_summaries[100])

Original Text: it can be any color, and any type, as long as it could not potentially harm your guinea pig. set it in the a
rea you will be training in. be sure to turn it upside-down, so that your guinea pig will not knock it over as you begin to
teach the trick. a garbage can that may have contained anything that could be harmful should not be used. , gently lift the
m out of their cage and set them down near the garbage can/other object. allow the guinea pig to sniff around if they feel
the need. some may ignore it, but this does not mean that they will not do the trick-have patience. , hold out one of the t
reats and lure your guinea pig towards the garbage can. move it slightly, further away and around it, until your guinea pig
has completed a quarter of the circle. click and treat. do this three or four times, until your guinea pig becomes comforta
ble with moving around it a bit. , every time the guinea pig moves forward a few steps, reward them with a click(or verbal
marker) and a treat. gradually increase the amount of steps they take before receiving the treat. hopefully, your guinea pi
g will soon discover that moving around the garbage can does not go without rewards. many will figure this out within a few
minutes of doing that same thing, but if they don't, go back to luring and go on from there as many times as your guinea pi
g needs it. , some guinea pigs will pick this up faster than others, and that is okay. no piggy is perfect, no matter how i
ntelligent they may be. , the hand motion can be whatever you see fit, but i have found that holding a finger above the obj
ect, where the guinea pig can clearly see it, and moving it around in a circular motion. saying the command "around!" or "s
pin!" may help as well, but verbal commands are more for the trainer than for the animal themselves. from this point on, on
ly treat your guinea pig when they walk around after your hand motion. if they do it without your motion, just sit patientl
y until they are done and ask them to do it afterwards. if they do it, treat and praise them immensely. it takes many guine
a pigs a long time to understand this, but, if the trainer is persistent, it will be very rewarding in the end. , once the
guinea pig is very good at walking around the garbage can on your hand motion, put the can up somewhere and do the motion.
this can be very easy, or very hard, depending on the piggy.

Human Summary: get a small garbage can or round, cylinder-shaped object of some sort., place your guinea pig in the general
area., use a treat to lure your guinea pig to it., stand back and see if the guinea pig will try to go around on its own.,
eventually, begin only treating your guinea pig when they make a full circle around the object., begin enforcing a hand mot
ion., take the garbage can away gradually.

Lsa Summarizer: from this point on , only treat your guinea pig when they walk around after your hand motion .

ROUGE-1: 0.23809523809523808
ROUGE-2: 0.1
ROUGE-L: 0.23809523809523808
BLEU Score: 0.02484051982038818
```

Figure 4.2: WikiHow TextRank result

```
Human Summary: get a small garbage can or round, cylinder-shaped object of some sort., place your guinea pig in the general
area., use a treat to lure your guinea pig to it., stand back and see if the guinea pig will try to go around on its own.,
eventually, begin only treating your guinea pig when they make a full circle around the object., begin enforcing a hand mot
ion., take the garbage can away gradually.

TextRank Summarizer: from this point on , only treat your guinea pig when they walk around after your hand motion .

ROUGE-1: 0.23809523809523808
ROUGE-2: 0.1
ROUGE-L: 0.23809523809523808
BLEU Score: 0.02484051982038818
```

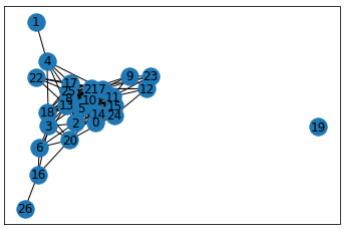


Figure 4.3: WikiHow LexRank result

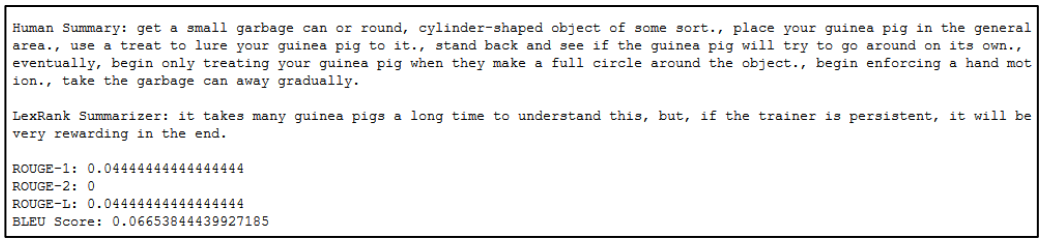


Figure 4.4: WikiHow Luhn Algorithm result

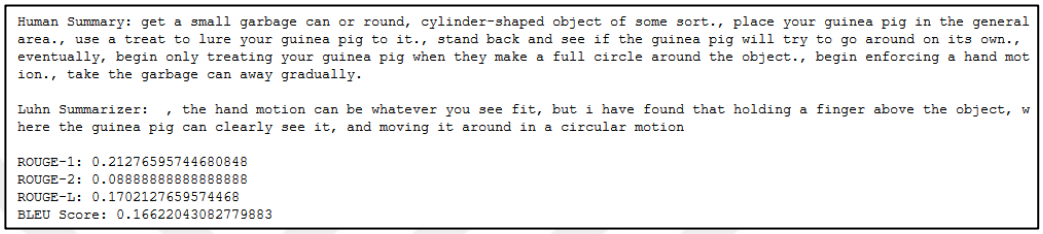
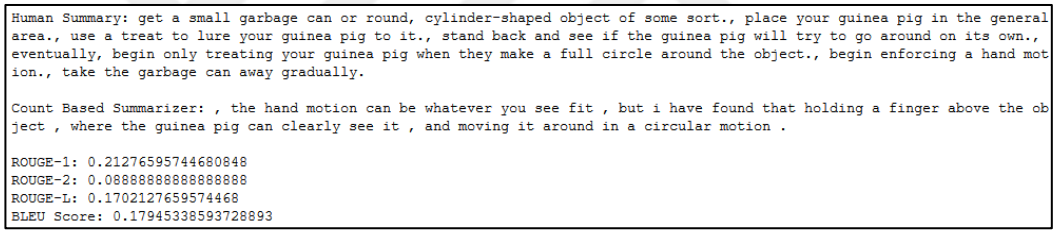


Figure 4.5: WikiHow CountBased result



4.2.2 Amazon Fine Food Reviews Results

Figures which are shown at the below shows results of the Amazon Fine Food Reviews dataset.

Figure 4.6: Amazon Fine Food Reviews LSA result

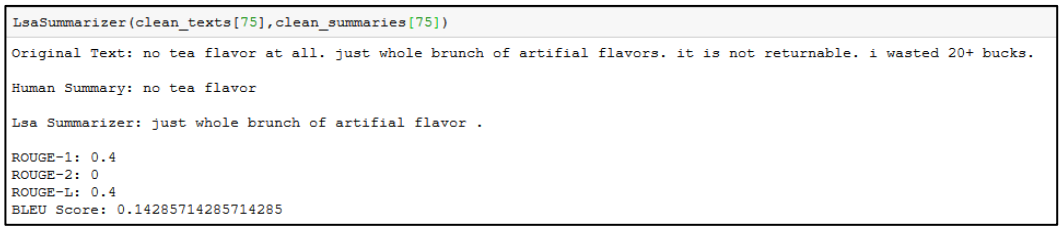


Figure 4.7: Amazon Fine Food Reviews TextRank result

```
TextRankSummarizer(clean_texts[75],clean_summaries[75])

Original Text: no tea flavor at all. just whole brunch of artifial flavors. it is not returnable. i wasted 20+ bucks.

Human Summary: no tea flavor

TextRank Summarizer: just whole brunch of artifial flavor .

ROUGE-1: 0.4
ROUGE-2: 0
ROUGE-L: 0.4
BLEU Score: 0.14285714285714285
```

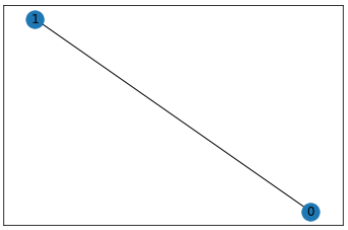


Figure 4.8: Amazon Fine Food Reviews LexRank result

```
LexRankSummarizerSumy(clean_texts[75],clean_summaries[75])

Original Text: no tea flavor at all. just whole brunch of artifial flavors. it is not returnable. i wasted 20+ bucks.

Human Summary: no tea flavor

LexRank Summarizer: no tea flavor at all.

ROUGE-1: 1.0
ROUGE-2: 1.0
ROUGE-L: 1.0
BLEU Score: 0.5
```

Figure 4.9: Amazon Fine Food Reviews Luhn Algoritmh result

```
Original Text: watch your prices with this. while the assortment was good, and i did get this on a gold box purchase, the p
rice for this was$3-4 less at target.

Human Summary: better price for this at target

Luhn Summarizer:  while the assortment was good, and i did get this on a gold box purchase, the price for this was$3-4 less
at target

ROUGE-1: 0.3636363636363636
ROUGE-2: 0
ROUGE-L: 0.3636363636363636
BLEU Score: 2.088284391396224le-78
```

Figure 4.10: Amazon Fine Food Reviews CountBased result

```
CountBasedSummarizer(clean_texts[75],clean_summaries[75])

Original Text: no tea flavor at all. just whole brunch of artifial flavors. it is not returnable. i wasted 20+ bucks.

Human Summary: no tea flavor

Count Based Summarizer: just whole brunch of artifial flavor .

ROUGE-1: 0.4
ROUGE-2: 0
ROUGE-L: 0.4
BLEU Score: 0.14285714285714285
```

4.2.3 News Summary Results

Figures which are shown at the below shows results of the News Summary dataset.

Figure 4.11: News Summary LSA result

```
LsaSummarizer(clean_texts[65],clean_summaries[65])

Original Text: airtel on wednesday said that qatar's sovereign wealth fund qatar investment authority will invest $200 mill
ion through a primary equity issuance in airtel africa. india's second-largest telecom operator's africa unit recently rais
ed $1.25 billion from six investors. airtel africa, the holding company for airtel's operations in 14 african countries, is
preparing for an initial public offering.

Human Summary: qatar to invest $200 million in airtel africa

Lsa Summarizer: airtel africa , the holding company for airtel 's operation in 14 african country , is preparing for an ini
tial public offering .

ROUGE-1: 0.21052631578947367
ROUGE-2: 0.11764705882352941
ROUGE-L: 0.21052631578947367
BLEU Score: 0.13043478260869565
```

Figure 4.12: News Summary TextRank result

```
Human Summary: qatar to invest $200 million in airtel africa

TextRank Summarizer: airtel africa , the holding company for airtel 's operation in 14 african country , is preparing for a
n initial public offering .

ROUGE-1: 0.21052631578947367
ROUGE-2: 0.11764705882352941
ROUGE-L: 0.21052631578947367
BLEU Score: 0.13043478260869565
```

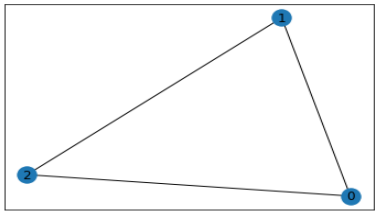


Figure 4.13: News Summary LexRank result

```
Human Summary: qatar to invest $200 million in airtel africa

LexRank Summarizer: airtel on wednesday said that qatar's sovereign wealth fund qatar investment authority will invest $200
million through a primary equity issuance in airtel africa.

ROUGE-1: 0.5217391304347826
ROUGE-2: 0.2857142857142857
ROUGE-L: 0.5217391304347826
BLEU Score: 0.2962962962962963
```

Figure 4.14: News Summary Luhn Algorithm result

```
Human Summary: qatar to invest $200 million in airtel africa

Luhn Summarizer: airtel on wednesday said that qatar's sovereign wealth fund qatar investment authority will invest $200 mi
llion through a primary equity issuance in airtel africa

ROUGE-1: 0.5217391304347826
ROUGE-2: 0.2857142857142857
ROUGE-L: 0.5217391304347826
BLEU Score: 0.13523285066501545
```

Figure 4.15: News Summary CountBased result

```
Human Summary: qatar to invest $200 million in airtel africa

Count Based Summarizer: airtel on wednesday said that qatar 's sovereign wealth fund qatar investment authority will invest
$ 200 million through a primary equity issuance in airtel africa .

ROUGE-1: 0.5217391304347826
ROUGE-2: 0.2857142857142857
ROUGE-L: 0.5217391304347826
BLEU Score: 0.2962962962962963
```

4.3 EXTRACTIVE METHODS SUMMARY EVALUATION

In the WikiHow dataset, the average score table as a result of the 10000 summary attempts is as follows **Table 4.1**.

Table 4.1: Extractive methods WikiHow results

WikiHow	Rouge 1	Rouge 2	Rouge L	Bleu
LSA	0.126	0.021	0.11	0.11
TextRank	0.132	0.022	0.15	0.12
LexRank	0.126	0.020	0.104	0.135
Luhn	0.106	0.017	0.090	0.108
CountBased	0.127	0.019	0.107	0.145

Scoring calculation was done by scoring function for each algorithm. When we examine the **Table 4.1** for the WikiHow dataset, we see that the scores are very close to each other. WikiHow dataset with the most words in terms of text size compared to other dataset. Rouge 1 has the highest score in the **TextRank** algorithm. As mentioned, **Rouge 1** is scoring on the basis of the word overlap. In this case we can say that TextRank gives better results for large corpus. Similarly, the LSA and TextRank algorithms in the Rouge 2 values seem to have received high scores. Rouge 2 bigram is the evaluation of binary word groups. According to Rouge 1 and Rouge 2 results, TextRank algorithm is very successful in large corpus. Rouge L is measuring the longest matching word order. In Rouge L rating, we see that TextRank gets the highest score. We understand that TextRank can make more accurate sentence selections as the reference weight given to sentences. Similarly, in LSA, the word count seems to be quite successful in the Rouge L evaluation. The Bleu evaluation seems to have received fairly low results for all extractive text summarization algorithms. In this case we can say that choosing the best sentence from text is far from the human summarization.

Table 4.2: Extractive methods Amazon Fine Food Reviews results

Amazon FR	Rouge 1	Rouge 2	Rouge L	Bleu
LSA	0.105	0.022	0.101	0.065
TextRank	0.088	0.016	0.084	0.059
LexRank	0.137	0.036	0.132	0.079
Luhn	0.096	0.023	0.092	0.055
CountBased	0.090	0.018	0.086	0.053

The Amazon Fine Food Reviews dataset was similarly evaluated with the WikiHow dataset. The Amazon Fine Food Reviews dataset consists of shorter text and related abstracts. In other words, we can evaluate the performance of algorithms on short texts. When we look at the scores which are shown at the **Table 4.2**, we see that LexRank has a significantly higher score than Rouge 1, Rouge 2 and Rouge L points. The similarity formulas, such as cosine similarity, used in the LexRank algorithm seem to be quite successful in the short summaries. In the Bleu scores, we can say that the algorithms closest to human abstracts are LSA and LexRank.

Table 4.3: Extractive methods News Summary results

News Sum.	Rouge 1	Rouge 2	Rouge L	Bleu
LSA	0.283	0.088	0.237	0.173
TextRank	0.255	0.076	0.214	0.158
LexRank	0.437	0.181	0.359	0.249
Luhn	0.378	0.146	0.310	0.227
CountBased	0.343	0.118	0.279	0.199

Same procedures applied on News Summary like the other ones. News Summary is a difficult dataset with spelling mistakes with non-standard grammatical errors. According to the Rouge 1 results which shown at **Table 4.3**, the highest score seems to have received the LexRank method. After that, Luhn algorithm quite successful, was an algorithm which measured the distance between the important keywords determined according to the frequency of the word. Rouge 2 values are very low for all algorithms. LexRank, has been rated higher than on Rouge 2 and Rouge L. We can say that all algorithms have low scores in the bleu evaluation. They look like they are not getting close to human approach.

Table 4.4: Overall results

Overall	Rouge 1	Rouge 2	Rouge L	Bleu
LSA	0,132	0,023	0,149	0,116
TextRank	0,159	0,038	0,149	0,112
LexRank	0,233	0,079	0,198	0,166
Luhn	0,193	0,062	0,164	0,113
CountBased	0,186	0,051	0,157	0,245

The overall evaluation was based on the average of 3 tables above **Table 4.4**. According to this assessment, the most successful algorithm in Rouge 1, Rouge 2 and Rouge L rating is LexRank. The algorithms closest to the human outline are CountBased and LexRank.

4.4 ABSTRACTIVE METHODS RESULTS

Seq2Seq GRU architecture was tested with 3 different dataset 20 epoch numbers in the abstractive text summarization application. The **Google Colab** environment was used as the development and testing environment. The application was run on the GPU.

4.4.1 WikiHow Results

Following 5 table show the results of WikiHow results.

Table 4.5: Abstractive methods WikiHow Test 1

Test 1	pay close attention to any articles or advertisements that mention casinos. note the location that is mentioned in each article or advertisement that involves a casino. if no locations are mentioned, note any additional contact information, such as a website or phone number. use that information to find out where the casinos are. , if you learn about more than 1 casino in your local newspapers, use the internet to search the distance between your location and each casino. sites such as maps.google.com or mapquest.com will help you in this search.
Original Text	
Human Summary	use the internet to see a location of the city you are looking for
AI Summary	a good dog food
Rouge 1	0.190
Rouge 2	0
Rouge L	0.190
Bleu	0.100

Table 4.6: Abstractive methods WikiHow Test 2

Test 2	monkeys are in the order of primates, which is divided into families of new world primates (smaller, tree-dwelling monkeys from south america) and families of old world primates (larger, ground and tree-dwelling monkeys from asia and africa). each type of monkey has unique characteristics that affect how they fare as pets. before you decide which type to get, conduct a lot of research on its characteristics. read books, talk to owners, and get to know as many monkeys as you can in person. squirrel monkeys, capuchins, spiders, and macaques are all types of monkeys people keep as pets. some are more social, while others can be on the nervous side. these monkeys all have the same general care requirements, though larger monkeys will need more space than the smaller ones. apes, such a chimpanzees, gibbons and orangutans are not
Original Text	
Human Summary	research monkey species., be ready for a huge commitment., find out if it is legal to own a monkey in your state., find a dealer or breeder with an excellent reputation., find an exotic animals veterinarian nearby., build or buy a monkey cage., monkey-proof your house., keep the monkey's space clean., provide clean water bottles every day., feed him monkey biscuits, fruits and vegetables., give the monkey plenty of toys and stimulation., play with the monkey every day., never punish a monkey., do not let the monkey play with strangers., have fun communicating with your monkey.
AI Summary	buy a cage for your rabbit buy a cage for your rabbit buy a cage for your rabbit make sure your cage is safe for your rabbit to get your pet ready for your pet to get your rabbit in a cage or a bowl of water and a small bowl of water and a small bowl of water and a small bowl of water and a small bowl of water to make your own cage make sure your cage is safe to use your rabbit to get your rabbit in a cage that is not in your cage make sure your cage is safe to get your rabbit in a cage that is not in your cage and will be used to make sure your rabbit is not a good idea to provide your rabbit with a cage that is not to feed your rabbit a lot of food in your cage with a cage that is not too much in your area that you are going to use for
Rouge 1	0.033
Rouge 2	0
Rouge L	0.033
Bleu	0.128

Table 4.7: Abstractive methods WikiHow Test 3

Test 3	a toothbrush should do just the trick. get the bristles sudsy and get all the cracks and crevasses. yes, this job may have been time consuming and work, but a simple cleaning with some saddle soap and a sponge, even just once a month, can make all the difference. , a saddle shine and conditioner will make the leather supple, and look like it did when you bought it.
Original Text	
Human Summary	soak a sponge in a bucket of warm water., wring out the sponge till damp and wipe your saddle and tack(this opens the pores of the leather in the saddle to thoroughly clean and condition your tack when saddle leather soap is applied.), rub the sponge onto some saddle soap to get it nice and sudsy. , work in the soap into the leather from top to bottom,underneath and everywhere in-between.this may take some elbow grease and you might get a sore hand but the finished result is a nice supple saddle fit for a show. , use a dry rag to wipe all the suds off the saddle. , still some hard pressed dirt in there?, since you are already covered in suds and want your saddle to look brand new, why not shine it up a bit., you can usually
AI Summary	use a soft toothbrush to clean the area of the shoes and then rinse the shoes with warm water and a soft cloth or cloth to clean the shoes and then rinse the shoes with warm water and then rinse the shoes with warm water and then rinse the shoes with warm water and then rinse the shoes with warm water and rinse the shoes with warm water and a soft cloth or cloth to dry the shoes and then rinse the shoes with warm water and rinse the shoes with warm water and rinse the shoes with warm water and rinse the shoes with warm water and a soft cloth or cloth to dry the shoes and then rinse the shoes with warm water and rinse the shoes with warm water and rinse the shoes with warm water and a soft cloth or cloth to
Rouge 1	0.064
Rouge 2	0.013
Rouge L	0.064
Bleu	0.221

Table 4.8: Abstractive methods WikiHow Test 4

Test 4	put your saddle on a sawhorse, saddle rack, or something similar to hold it up while you clean it. this keeps your saddle stable and easy to adjust while you clean. once your saddle is stable, remove any add-ons such as stirrups. these should be cleaned separately.; , getting loose dirt off before wetting your saddle makes the rest of the cleaning faster and easier. use a dry cloth to remove any loose dirt and dust before wetting your saddle.avoid grinding dirt into the saddle while dusting. use a gentle hand to swipe over the surface of the saddle. be sure to use a clean cloth or soft-bristled saddle brush with nothing on it. , dip a double-sided kitchen sponge into a bucket of warm water. gently wring it out and lather it up with saddle soap. be sure the sponge is damp enough to create your lather, but not dripping wet. excess water may cause the
Original Text	
Human Summary	stabilize your saddle., dust your saddle., prepare a lather., soap your saddle., rinse the saddle., dry your saddle., condition the smooth leather., condition any rawhide areas., remove excess conditioner., decide if it is time to clean your saddle., check with the manufacturer., gather your supplies.
AI Summary	clean your horse s coat with warm water apply a coat of leather oil to your horse s coat apply a coat of leather oil to your horse s coat apply a coat of leather oil to your horse s coat apply a coat of leather oil to your horse s coat with a clean towel
Rouge 1	0.095
Rouge 2	0
Rouge L	0.095
Bleu	0.157

Table 4.9: Abstractive methods WikiHow Test 5

Test 5	while saddle soap can be great on many leather shoes, it can be harmful to some. cordovan shoes should not be shined or washed using saddle soap. if you are uncertain about using saddle shoes on your footwear, consult the manufacturer's instructions., snow, water, and salt can cause significant damage to leather. saddle soap can be used as a protective agent on leather boots. if you've recently worn your boots, you will need to let them dry before using saddle soap. do not attempt to dry your boots by setting them next to a heater. this can cause damage.wait for the boots to dry. this should take about an hour. using a damp cloth, remove any dirt or debris on the boots.take a wet cloth and rub it in saddle soap until it begins to lather. you do not want to rub saddle soap onto your boots until it lathers, so add more water if the soap remains as
Original Text	
Human Summary	stabilize your saddle., dust your saddle., prepare a lather., soap your saddle., rinse the saddle., dry your saddle., condition the smooth leather., condition any rawhide areas., remove excess conditioner., decide if it is time to clean your saddle., check with the manufacturer., gather your supplies.
AI Summary	avoid saddle soap on certain shoes., protect your boots for the winter months., clean your shoes with saddle soap
Rouge 1	0.275
Rouge 2	0.074
Rouge L	0.206
Bleu	0.153

4.4.2 Amazon Fine Food Reviews Results

Following 5 table show the results of Amazon Fine Food Reviews results.

Table 4.10: Abstractive methods Amazon Fine Food Reviews Test 1

Test 1	i have been very pleased with the natural balance dog food. our dogs have had issues with other dog foods in the past and i had someone recommend natural balance grain free since it is possible they were allergic to grains. since switching i have not had any issues. it is also helpful that have have different kibble size for larger smaller sized dogs.
Original Text	
Human Summary	good healthy dog food
AI Summary	wellness healthy dog food
Rouge 1	0.727
Rouge 2	0.666
Rouge L	0.727
Bleu	0.75

Table 4.11: Abstractive methods Amazon Fine Food Reviews Test 2

Test 2	we have three dogs and all of them love this food! we bought it specifically for one of our dogs who has food allergies and it works great for him, no more hot spots or tummy problems.i love that it ships right to our door with free shipping.
Original Text	
Human Summary	great food!
AI Summary	great food
Rouge 1	0.857
Rouge 2	0.8
Rouge L	0.857
Bleu	0.606

Table 4.12: Abstractive methods Amazon Fine Food Reviews Test 3

Test 3	these taste really good. i have been purchasing a different brand and these are very similar in taste and texture. i agree with the other reviewer regarding ordering in the summer. there is no insulating packaging with ice packs so they will melt in warm weather like all chocolate food items. order in cold weather and buy enough to last!!!
Original Text	
Human Summary	taste great
AI Summary	great taste
Rouge 1	0.857
Rouge 2	0
Rouge L	0.571
Bleu	1.0

Table 4.13: Abstractive methods Amazon Fine Food Reviews Test 4

Test 4	buyer beware please! this sweetener is not for everybody. maltitol is an alcohol sugar and can be undigestible in the body. you will know a short time after consuming it if you are one of the unsuspecting many who cannot digest it by the extreme intestinal bloating and cramping and massive amounts of gas a person can experience. nausea, diarrhea & headaches can also be experienced. i learned my lesson the hard way years ago when i fell in love with the sugar-free chocolates suzanne sommers used to sell. i thought i would found sugar-free chocolate nirvana at first taste but the bliss was short lived when
Original Text	
Human Summary	warning! warning! -alcohol sugars!
AI Summary	warning warning warning read this product
Rouge 1	0.5
Rouge 2	0.2
Rouge L	0.5
Bleu	0.282

Table 4.14: Abstractive methods Amazon Fine Food Reviews Test 5

Test 5	i have been drinking royal king 100% natural organic green tea (100 tea bags x 2g each) as my every day tea for several years now. i buy 12 boxes at a time to save on shipping. for many years i used to drink coffee from morning till night. but finally i realized that drinking so much coffee was not healthy for me. i finally resolved to improve my health and stopped drinking coffee all together. i tried many alternative drinks to replace my coffee habit. i found that green tea was not only good tasting it also has health benefits. green tea is one of the few drinks that actually makes you healthier (coffe and other
Original Text	
Human Summary	my favorite green tea
AI Summary	my every day green tea
Rouge 1	0.666
Rouge 2	0.571
Rouge L	0.666
Bleu	0.584

4.4.3 News Summary Results

Following 5 table show the results of News Summary results.

Table 4.15: Abstractive methods News Summary Test 1

Test 1	gabrielle reilly, a 22-year-old us woman, created earrings for her apple airpods and put it up for sale online for $20 (about 1,500). "i absolutely refuse to lose them (airpods)...so i made earrings," she explained. the earrings, which took her over an hour to make, debuted in a video on twitter that has since garnered over three million views.
Original Text	
Human Summary	woman turns apple airpods into earrings to avoid losing them
AI Summary	woman turns back over apple from selling iphone x
Rouge 1	0.5
Rouge 2	0.142
Rouge L	0.5
Bleu	0.298

Table 4.16: Abstractive methods News Summary Test 2

Test 2	twenty-seven-year-old mohammed mahuwala was arrested in indore on wednesday for allegedly cheating e-commerce giant amazon of nearly 30 lakh. mahuwala was a member of a gang who ordered costly gadgets from amazon. " they used to get refund of the amount paid...by saying the parcel...was empty. in reality, these devices were taken out from parcel and sold...to local shopkeepers," said police.
Original Text	
Human Summary	man arrested for cheating amazon of 30 lakh by taking refunds
AI Summary	man arrested for stealing 500 crore worth at bitcoin
Rouge 1	0.333
Rouge 2	0.125
Rouge L	0.333
Bleu	0.271

Table 4.17: Abstractive methods News Summary Test 3

Test 3	german multinational engineering and electronics firm bosch has made its first investment in india in bengaluru-based deep-tech startup simyog. simyog has raised about 6.3 crore in the funding round, with participation from early-stage venture capital firm ideaspring capital. incubated at the indian institute of science (iisc), it provides design and sign-off tools for automotive electronics.
Original Text	
Human Summary	bosch makes its 1st india investment in iisc spinoff simyog
AI Summary	startup in india first in 1st tech made in india raises 20 million
Rouge 1	0.3
Rouge 2	0
Rouge L	0.3
Bleu	0.230

Table 4.18: Abstractive methods News Summary Test 4

Test 4	railway police has rescued a woman travelling on a train in chennai who got her leg stuck inside the commode of a toilet. after being caught inside the commode hole, she struggled to remove her leg and following failed attempts, she shouted for help, reports said. railway police used a cutter to pry open the commode and freed her leg.
Original Text	
Human Summary	railway police rescues woman with leg stuck in train toilet
AI Summary	railway woman gets stuck in train toilet for a day of woman
Rouge 1	0.666
Rouge 2	0.25
Rouge L	0.666
Bleu	0.5

Table 4.19: Abstractive methods News Summary Test 5

Test 5	the gujarat government has asked ride-hailing services like ola and uber to start electric vehicle services. the state energy department, led by gujarat's energy minister saurabh patel, is working on creating a policy to provide required infrastructure to mass-run electric vehicles. patel said recharge stations with charged batteries will be available in a manner similar to existing petrol pumps.
Original Text	
Human Summary	gujarat govt asks ola, uber to start electric taxi services
AI Summary	ola uber plans to launch electric vehicles in delhi
Rouge 1	0.421
Rouge 2	0.117
Rouge L	0.421
Bleu	0.361

4.5 ABSTRACTIVE METHODS SUMMARY EVALUATION

After 10000 test iteration, results for WikiHow dataset are as follows. 10000 different text was selected from the dataset for evaluation.

Table 4.20: Abstractive Methods WikiHow Results

WikiHow	Rouge 1	Rouge 2	Rouge L	Bleu
10000	0.091	0.008	0.061	0.079

When we review the results of the WikiHow which is shown at **Table 4.20**, Generally low scores have been taken. Rouge scores are almost close to extractive summarization methods. Seq2Seq GRU system has made repetitions of words as seen in the summary sentences. The fact that the WikiHow dataset consists of quite large text. According to the results of the 20 epoch training, the dataset needs to go through a lot more training. 20 Epoch training took more than 20 hours in the Google colab environment. One way to increase the success rate for this dataset can be to increase the amount of training. In addition, a larger word embedding matrix can be created. Different Rnn types such as LSTM can be tried. We can increase the accuracy rate by adding BRNN.

Results for the Amazon Fine Food Reviews dataset are as follows **Table 4.21**. 10000 different texts were selected from the dataset for evaluation.

Table 4.21: Abstractive Methods Amazon Fine Food Review Results

Amazon FR	Rouge 1	Rouge 2	Rouge L	Bleu
10000	0.348	0.198	0.347	0.219

When we look at the results of Amazon Fine Food Reviews, we see very good results in general.We can say short text and summaries increase the accuracy rates. Among the Bleu results, we see that the highest score is taken as a result of 1. This result shows that the same result is estimated with the complete human outline. When we compare the results of the rouge with the extractive algorithms, we can say that the results are good. In this dataset, similarly, 20 epoch were run. Training time took about 4 hours.

Results over 10000 iterations for the News Summary dataset are as follows **Table 4.22**. 10000 different texts were selected from the dataset for evaluation.

Table 4.22: Abstractive Methods News Summary Results

News Sum.	Rouge 1	Rouge 2	Rouge L	Bleu
10000	0.279	0.073	0.262	0.265

When we review the News Summary results, we can say that satisfactory scores have been obtained, although not as good as Amazon Fine Food Reviews. The news summary is a dataset that contains mostly spelling and grammatical errors. Despite this, the Bleu scores seem to be on average. If we compare the results with the Extractive summarization, we see that the Rouge scores are close. News summary 20 epoch was trained and lasted for about 1 hour. Due to the fact that it can be trained quickly, we can increase the number of training considerably.

5. CONCLUSION AND FUTURE WORKS

Within the scope of the thesis, automatic text summarization methods were investigated. Extractive and abstractive text summarization algorithms have been processed theoretically and practically. Differences between algorithms were mentioned.

When we look at the results, the first thing that caught my attention was that the extractive algorithms were very close to each other and reached a satisfaction point in the following years. Nevertheless, I have observed that it works very well, especially when it comes to large texts. Extractive applications work very fast. Choosing the most important sentence from a text really allows us to focus on the subject. However, the disadvantage is that it remains far from human summaries.

Abstractive text summation methods take quite some time to develop due to their rather difficult architectural structures. It requires powerful equipment to be trained on large data. It is quite a problem to train an application using a GRU-based Seq2Seq structure on a personal computer. I have observed that even 1 Epoch training cannot be completed on my personal computer. Thanks to the Google Colab environment, this problem can be partially overcome. Despite these challenges, I can say that deep learning and nlp issues promise a great future. As I observed in the abstractive text summarization tests, in the case of large texts, some words become repetitive. That's a pretty big problem. GRU enables rapid development, but we can say that it is not enough alone.

Adding BRNN to GRU structures designed as Seq2Seq Encoder and Decoder seems to be quite a logical choice. In this case, our training period will be longer, but we can get better results. In addition, the recently popular Attention mechanism is a method that should be tested, especially in the Context Vector, which allows us to choose the most appropriate probability of right word. I think that very satisfactory results can be obtained with BRNN and Attention on GRU or LSTM.

Tensorflow and Keras api, the artificial intelligence framework of Google, enable faster applications in the industry. Thanks to Google's tensorflow, Facebook's pytorch framework, or Microsoft's ML.net, I think that the artificial intelligence industry will have serious breakthroughs in the years ahead.

REFERENCES

Books

Minsky, M.,1986. *Society of minds*. Simond & Schuster

Sarkar, D.,2016. *Text analytics with python*. Apress

Periodicals

Turing, A., 1950. Computing Machinery and Intelligence. *Oxford University Press on behalf of the Mind Association.* **59** (236), pp.433-460

Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**(6), pp.386-408.

Hinton, G., 1986. The perceptron: Learning internal representations by error-propagation. *MIT Press, Cambridge,* MA, **1**(6088), pp.318-362.

Yogan,. Kumar J., Goh., Sing O,. Halizah., Basiron., Ngo., Choon H., Puspalata., Suppiah C., 2016. A Review On Automatic Text Summarization Approaches. *Journal Of Computer Science*, **12** (4), pp.178-190.

Robertson, S., 2004. Understanding Inverse Document Frequency: On theoretical arguments for IDF, *Journal of Documentation*, **60**(5), pp.503-520.

Landauer,TK., Thomas K.,Dumais., Susan T., 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, **104**(2), pp.211-240.

Ozsoy, M., Alpaslan, F., Cicekli, I., 2011. Text summarization using Latent Semantic Analysis, *Journal of Information Science*, **37**(4), pp.405-417.

Steinberger, J., Jezek, K., 2004. Using Latent Semantic Analysis in Text Summarization and Summary Evaluation. *Proceedings of ISIM '04*, pp. 93-100.

Gong, Y., Liu, X., 2001. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. In: Proceedings of the 24th ACM SIGIR conference on Research and development in information retrieval, New Orleans, Louisiana, United States, pp. 19–25

Page, L., Brin, S., 1998. The anatomy of a large-scale hypertextual Web search engine, *Computer Networks and ISDN Systems*, **30**(1-7), pp.107-117.

Erkan, G., Radev, D., 2004. LexRank:Graph-basedLexicalCentrality as Salience in Text Summarization, *Journal Of Artificial Intelligence Research*, pp. 1724-1734.

Luhn, HP., 1958. The automatic creation of literature abstracts, *IBM Journal of research and development*, **2**(2), pp. 159–165.

Cho, K., van Merrienboer, B., van Merrienboer, B., Gulcehre,C., Gulcehre,C., Bahdanau,D., Bougares,F., Schwenk,H., Schwenk,H., Bengio,Y., 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, *Association for Computational Linguistics*, pp. 159–165.

Hochreiter, S., 1997. Long Short-Term Memory, *Neural Computation*, **9**(8), pp. 1735-1780 .

Other Publications

Mihalcea, R., Tarau, P., 2004. Bringing Order into Texts, *In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona,* Spain, 25–26 July 2004, pp. 404–411

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient Estimation of Word Representations in Vector Space [online] https://arxiv.org/abs/1301.3781 [Accessed 3 May 2019]

Mikolov, T., Sutskever, I., Chen, K., Corrado, H., Dean, J., 2013. Distributed Representations of Words and Phrases and their Compositionality [online] https://arxiv.org/pdf/1412.3555.pdf [Accessed 3 May 2019]

Koupaee, M., Wang, Y,W., 2018. Distributed Representations of Words and Phrases and their Compositionality [online] https://arxiv.org/pdf/1412.3555.pdf [Accessed 3 May 2019]

McAuley, J., 2014. Amazon product data [online] http://jmcauley.ucsd.edu/data/amazon/ [Accessed 3 May 2019]

Zhang, D., 2006. PageRank Examples [online] http://www.dcs.bbk.ac.uk/~dell/teaching/ir/examples/pr_example.pdf [Accessed 3 May 2019]

Murray, G., Renals, S., Carletta, J., Moore, J., 2005. Evaluating Automatic Summaries of Meeting Recordings [online] https://www.era.lib.ed.ac.uk/handle/1842/1040 [Accessed 3 May 2019]

Chung, J., Gulcehre, C., Cho, K., Bengio, Y., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling [online] https://arxiv.org/abs/1412.3555 [Accessed 3 May 2019]

Olah, C., 2015. Understanding LSTM Networks. [Online]: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ [accessed on 03 May 2019].

*Stemming and lemmatization.*2019.

https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html [03 May 2019]

*Linear regression.*2009. https://en.wikipedia.org/wiki/Linear_regression [03 May 2019]

*Logistic regression data classification.*20018. https://veribilimcisi.com/2017/07/18/lojistik-regresyon/ [03 May 2019]

*Sigmoid function.*2019. https://en.wikipedia.org/wiki/Sigmoid_function [03 May 2019]

*Gradient descent.*2018. https://hackernoon.com/gradient-descent-aynk-7cbe95a778da

[03 May 2019]

*Architecture of cnn.*2018. https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html [03 May 2009]

*Convolutional formula.*2018. http://mathworld.wolfram.com/Convolution.html [03 May 2019]

*Dertat,A.,Convolutional operation.*2018. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2 [03 May 2019]

*Cnn fully connected layer.*2018. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection [03 May 2009]

*Karpaty,A.,Rnn Types.*2015. http://karpathy.github.io/2015/05/21/rnn-effectiveness/ [03 May 2019]

Kostadinov, S,.Seq2seq architecture.2019. https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346 [03 May 2019]

Ng, A,.Coursera rnn,word representation course notes. 2019. https://www.coursera.org/lecture/nlp-sequence-models/recurrent-neural-network-model-ftkzt [03 May 2019]

Ng, A., Language model and sequence generation coursera course notes, 2019 https://www.coursera.org/lecture/nlp-sequence-models/language-model-and-sequence-generation-gw1Xw [03 May 2019]