ERDİNÇ AYSU

M.S. Thesis

2018

BIG DATA STORAGE AND AUTOMATED TEXT SUMMARIZATION IN TURKISH TEXT

ERDİNÇ AYSU

IŞIK UNIVERSITY
2018

BIG DATA STORAGE AND AUTOMATED TEXT SUMMARIZATION IN TURKISH TEXT

ERDİNÇ AYSU
B.S., Computer Engineering, IŞIK UNIVERSITY, 2012

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

IŞIK UNIVERSITY
2018

IŞIK UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

BIG DATA STORAGE AND AUTOMATED TEXT SUMMARIZATION IN TURKISH TEXT

ERDİNÇ AYSU

APPROVED BY:

Prof. Dr. Olcay Taner YILDIZ (Thesis Supervisor)	Işık University
Assist. Prof. Nilgün Güler BAYAZIT	Yıldız Technical University
Assist. Prof. Ayşegül TÜYSÜZ ERMAN	Işık University

APPROVAL DATE: 19/06/2018

BIG DATA STORAGE AND AUTOMATED TEXT SUMMARIZATION IN TURKISH TEXT

Abstract

The subject of this study is storing the large datasets in accordance with Big Data ecosystem and to extract the summary sentences of a text in Turkish, apply the automatic text summarization process which is a subtopic of Natural language processing (NLP). For this purpose, Turkish news articles were collected and the study was carried out through these texts. For the performance test of the work done, 50 different news textiles were given to 20 different persons and 3 sentences which were considered important from each other were asked to be selected and their results were compared with each other. Then, the results from the people were compared with the results from this study. As a result of the test process, the summation performance of the work was measured approximately as thirty-six percentage.

Keywords: Big Data, Hadoop, NLP, Summarization

DEV VERİ DEPOLAMA VE TÜRKÇE METİN İÇİN OTOMATİK ÖZETLEME

Özet

Bu çalışmanın konusu, geniş çapta veriyi Dev Veri ekosistemine uygun bir şekilde saklamak ve bir Türkçe dokumanın özet cümlelerinin çıkarılması için doğal dil işleme (DDİ) alt konusu olan otomatik metin özetleme işlemini uygulamaktır. Bu amaçla Türkçe haber metinleri toplanmış ve çalışma bu metinler üzerinden yürütülmüştür. Yapılan çalışmanın performans testi için 20 farklı kişiye 50 farklı haber metni verilmiş ve her metnin içerisinden önemli gördükleri 3 cümlenin seçilmesi istenmiştir ve sonuçlar birbirleriyle karşılaştırılmıştır. Daha sonra kişilerden alınan sonuç ile bu çalışmadaki çıkan sonuç karşılaştırılmıştır. Test işleminin neticesinde çalışmanın özetleme performansı yaklaşık olarak yüzde otuz altı ölçülmüştür.

Anahtar Kelimeler: Dev Veri, Hadoop, DDİ, Özetleme

Acknowledgments

I would like to thank all the masters who have shared their knowledge and experiences during my undergraduate and postgraduate studies to contribute to my personal development.

From the beginning to the end of this study, my valuable master who has shown patience and helped me to complete my study, I would also like special thanks to Prof. Olcay Taner Yıldız.

To MSB…

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1. Context

In the digital era that we are in, all the data in our lives are transferred rapidly from the real environment to the virtual environment, and as a result, the size of the data kept in the digital environment increases tremendously. With the comprehension of the importance of digital data, the information obtained from these data has created new horizons. In the light of these developments, new concepts and new techniques have emerged for storing and processing data. One of these developments is the formation of an ecosystem called Big Data and the processing of data held in this ecosystem with machine learning techniques has started a new era in the industry.

A digital data concept contains many components. The data referred to in this component may be in the form of photographs, video, audio, or text. In this study, we handle the data in the text form. We need a text summarization process. It takes a considerable amount of time to read or browse texts (books, articles, news, etc.) held in digital form due to the increased amount and length. Most of the work done in the summarization methodologies based on English. In this study, we will propose an automatic summarization of Turkish texts and it is based on the storage of the results on a distributed system.

1.2. The Aim of the Study

In this study, it is aimed to extract important documents of a Turkish language found in a digital environment and store them in a distributed system in the form of a summary file. Hadoop installation for this distributed system and use of HDFS for storage of data is targeted.

1.3. Thesis Overview

We discussed the purpose of the study in Chapter 1. We explain the Big Data, Hadoop and Summarization techniques literature in Chapter 2. We present distributed multi-node Hadoop system setup with configuration adjustment for experimental work in Chapter 3. We show the test of the study and the result of this test in Chapter 4, discuss future work in Chapter 5 and give a conclusion in Chapter 6.

Chapter 2

Literature Review

In this chapter, we tackle some fundamentals of the big data infrastructure, how to relate big data with NLP and explain summarization techniques.

2.1. Related Works

Ramesh Nallapati, Feifei Zhai_and Bowen Zhou explain their extractive summarization based approach in [2]. The approach based on Recurrent Neural Network (RNN) the sequence model for extractive summarization of documents. According to study, it allows sequence classifier that outperforms or matches state-of-the-art models for extractive summarization and the model presents a training mechanism that allows it to be trained end-to-end using abstractive summaries.

Rada Mihalcea and Paul Tarau show how to summarize the text using TextRank algorithm in their study [3]. The study relies on a graph-based ranking model which is derived from Google's PageRank (Brin and Page, 1998) model. According to TextRank model, the graph-based algorithm is a way that decides to the importance of a vertex looking to the whole graph recursively rather than relying only on local vertex-specific information. According to the model, the fundamental of keyword and sentence extraction is “voting” or “recommendation”.

Ebru UZUNDERE, Elda DEDJA, Banu DİRİ and M. Fatih AMASYALI studied Turkish text summarization on news article and gave the summarization methods at their study [4]. Their approach based on extractive sentences by order of importance from the news article text document. The selective methods rely on the weighted of sentences such as an entrance sentence of a paragraph or including keywords, number, and date.

The contribution of this study is extracting *action word* of the document which is based on prediction the predicate of sentences and chooses one of the most commonly used in the document. This *action word* helps to select candidate sentences to create a summary. The dataset and result from text documents have stored on distributed multi-node Hadoop cluster so that the dataset may be extendable.

2.2. A Brief Look to the Big Data

New digital world brings to pass new expressions like Big Data. Big Data means a huge amount of data which cannot be handled with traditional processes such as relational database management. It describes an ecosystem that includes a huge amount of data and new storage methods. Big Data is not just a collection of data, but also it includes new techniques about new data types, new storage systems.

According to International Data Corporation’s annual Digital Universe study, the size of the digital data is doubling every two years, and by 2020 the digital universe – the data we create and copy annually – will reach 44 zettabytes or 44 trillion gigabytes.

All developments in data collection and storage have opened up new inventions in the field of information technologies. As the industry realized the importance of Big Data and saw what it could do with it, it began to produce the products to adapt to the Big Data ecosystem. We can give many examples to the Internet of things practices which produce data (such as smart homes, connected car, wearables etc.). According to the Gartner, the connected IoT units will reach 20.4 billion by 2020 as shown in Table 2.1.

Table 2.1: IoT Units Installed Base by Category (Millions of Units) - Gartner (January 2017)

Category	**2016**	**2017**	**2018**	**2020**
Consumer	3,963.0	5,244.3	7,036.3	12,863.0
Business: Cross-Industry	1,102.1	1,501.0	2,132.6	4,381.4
Business: Vertical-Specific	1,316.6	1,635.4	2,027.7	3,171.0
Grand Total	**6,381.8**	**8,380.6**	**11,196.6**	**20,415.4**

2.2.1. Characteristics of Big Data

Big data has three well-known characteristics named as Volume, Variety, and Velocity (Figure 2.1.) Understanding these three "V"s will help to understand the capabilities of Big Data.

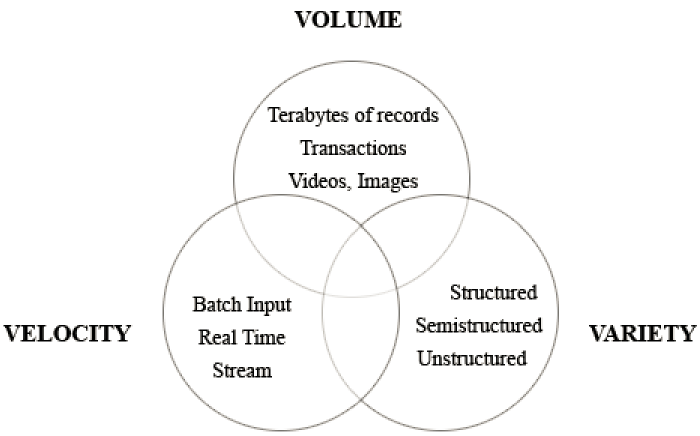


Figure 2.1: Characteristics of Big Data

2.2.1.1. Volume

Data is produced from everywhere even from a refrigerator, therefore the amount of data has increased in huge number in last years and it continues to grow. (Such that as a reported by a research from EMC in 2011 it is growing **40%** a year into the next decade shown in Figure 2.2.)

We can illustrate the increase in the number and quality of photos and videos held in social media. According to the study of Qmee, Twitter received 433K Tweets, Snapchat received 277K snaps, 25K items are purchased from Amazon.com only in 60 seconds in the year of 2014 as shown in Table 2.2.

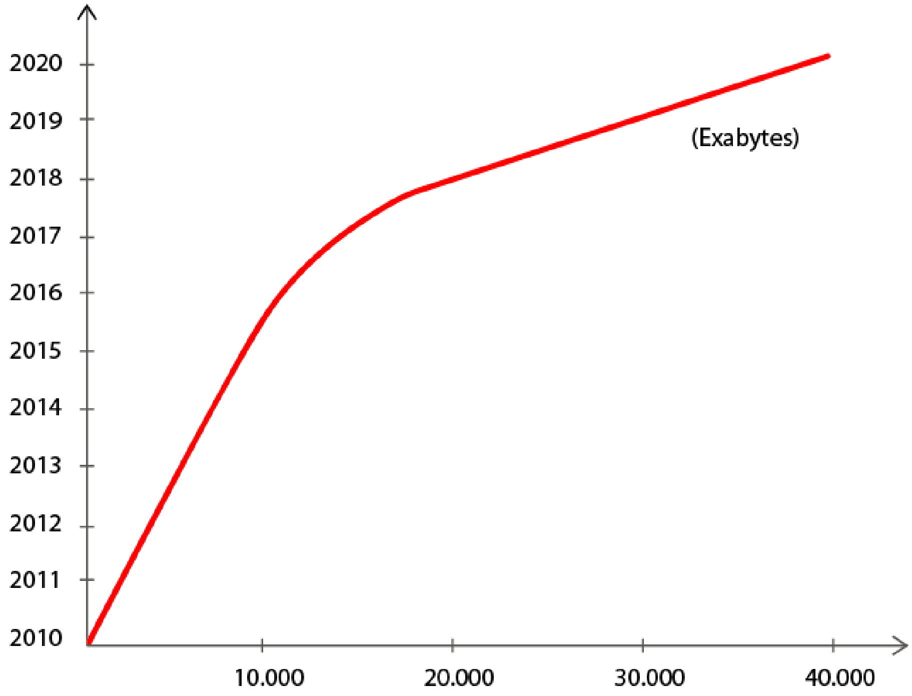


Figure 2.2: 50-fold Growth from the Beginning of 2010 to the End of 2020

Table 2.2: Data occurred in 60 seconds in the year of 2014 (Numbers from Qmee)

Source	**Amount Of Data Produced In 60 Seconds**
Youtube	Over 5 Million Videos Viewed
Facebook	293K Statuses are Updated
Twitter	433K Tweets
Skype	88K Calls
Vine	540 Vines
Google	2.66 Million Searches
Flickr	1.1K Photos Uploaded
Snapchat	277K Snaps
Amazon.com	25K Items Purchased
Linkedin	120 New Users
Pinterest	3.4K Pins
Instagram	67K Photos Uploaded
e-Mail	138.8 Million
Tumblr	4.7 Million Posts

2.2.1.2. Velocity

The huge increase of the data volume has negatively affected the data processing speed. The Velocity feature means that the system can handle this huge volume of data and can process the Big Data quickly with the advantage of distributed systems. With many techniques available in the Big Data ecosystem, this speed increase is achievable and can be optimized to cope with ever-increasing amounts of data.

The processing speed required by conventional data reading and writing techniques cannot be achieved in the relational database system. Big Data solves speed problem with in-memory systems. Many firms (such as SAP, Google etc) use this in-memory technique in their businesses.

2.2.1.3. Variety

We call the data types produced from heterogeneous sources structured, unstructured, and semi-structured. The variety characteristic of Big Data refers to the formation of that different data types.

It is known that many of the data on Earth are in non-structural form. Examples of non-structural data are photos, video and audio data generated from social media and data from sensors.

Data stored in the relational database are structured (RDBMS).

The output of recent applications of user experiences and internet clickstream records are examples of semi-structured data types.

2.2.2. Big Data Storage and Distributed Computing System

As the world digitizes, there is a need for massive storage space for the increasing amount of data and computational systems that can process this data efficiently. This system should provide high IOPS rate for Big Data applications such as real-time data analytics. Big data requires much larger datasets from traditional datasets. The storage capacity and analysis ability must not decrease with respect to the size of datasets.

Recently we see Flash storage (all flash or hybrid) systems in use on servers. That flash storage technology has benefits such as high-speed read and write I/O operations. It uses flash memory so ensure permanent memory even if the electricity cut off.

2.2.2.1. Distributed Calculation

Big Data needs big storage capacity and that capacity should be extendable. Using distributed systems increases the storage capacity. A distributed system is a system that allows multiple computers to connect to each other and communicate and operate as a whole. We can separate distributed systems into two the client-server and Peer-to-peer (P2P) systems which are shown in Figure 2.3 and Figure 2.4. respectively.

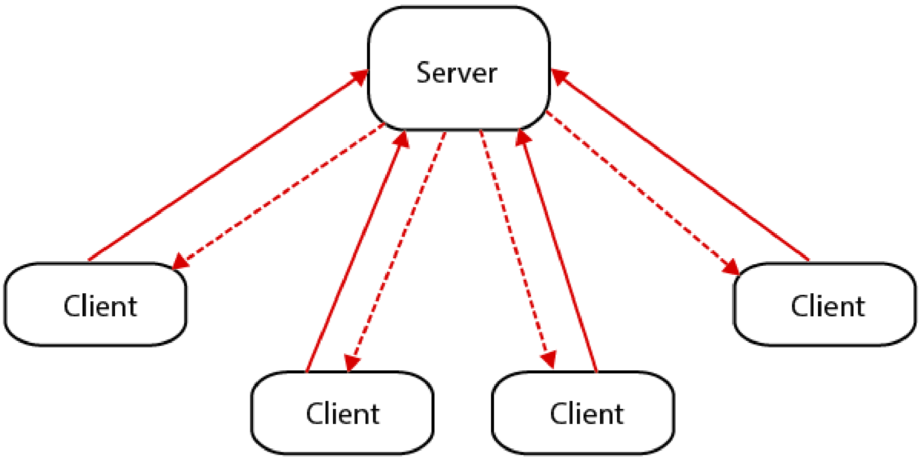


Figure 2.3: Client-server Distributed systems

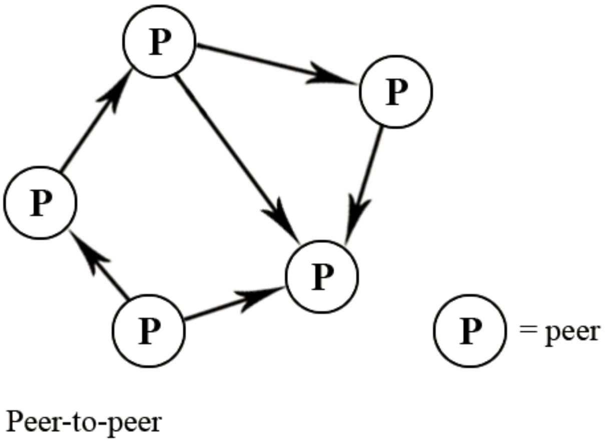


Figure 2.4: Peer-to-Peer Distributed systems

2.2.2.2. Relation of Machine Learning with Big Data

A growing number of applications are being built with machine learning tools that run on Big Data (such as Spotify, Graphflow, Zynga etc.)

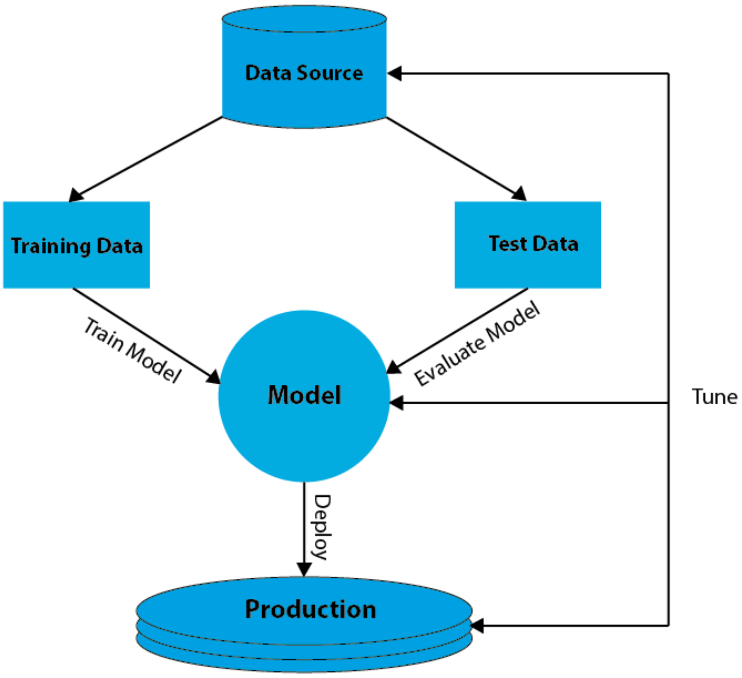


Figure 2.5: Supervised machine learning workflow

The goal of machine learning is to enable a system to learn from the past and use that knowledge to make predictions or decisions regarding unknown future events [5]. We see a workflow of a supervised machine learning system in Figure 2.5.

2.2.2.3. Computing Power

Computing resources have an essential role in the process in Big Data. Every single computer has a limit in terms of memory, disk size or processor speed. On the other hand, these limits are less important in a cluster that consists of multiple computers.

The main purpose of distributed systems is to ensure high computing power at low-cost using already existing computer materials. Instead of a high-cost, high computing power hardware, distributed systems use low-cost, low computing power hardware to gain desired computational power. Day by day the computing power grows even though the cost doesn’t grow.

2.2.2.4. Ideal Distributed Systems

To design an ideal distributed system it should has the crucial attributes listed in Figure 2.6. Accordingly, an ideal distributed system consist of features of Heterogeneity, Transparency, Concurrency, Scalability and Fault Tolerance.

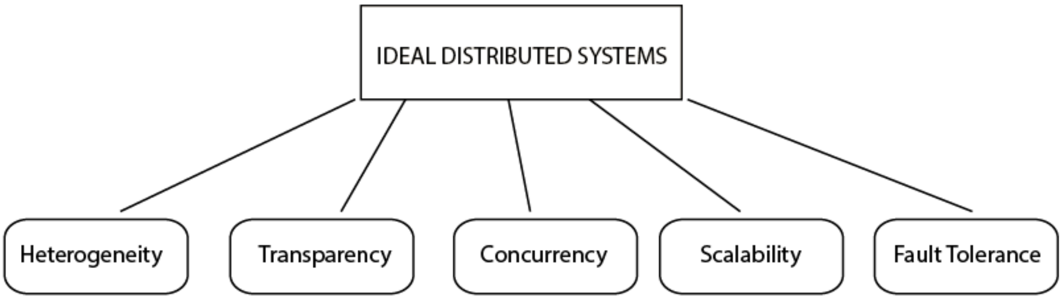


Figure 2.6: Ideal Distributed Systems

2.2.2.4.1. Scalability

The scalability is defined by B. Clifford Neuman as *"A system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity."*

2.3. Hadoop

In order to reduce the complexity of the computation and analysis processes that are created by ever-increasing data, the development of techniques such as Hadoop on distributed systems has opened up. Hadoop is an open-source Java-based programming framework that provides processing huge amount of datasets on multiple computing nodes.

Hadoop has a history that starts from 1999 with the announcement of Apache Software Foundation (ASF) and until today many companies used that distributed system framework library and caught the success such as Facebook, Yahoo, eBay, Linkedin, Amazon.

Hadoop tries to fix scalability problem using distributed storage and processing infrastructures and it ensures a very extensible platform that allows the creation of for many machine learning projects and applications (e.g. Mahout, RHadoop).

It could be said that the reason for the success of the Hadoop system is. The Hadoop system could store the massive volume of data in low-cost and analyze the data in a secure way. The Hadoop system one can add new nodes easy so that the system grows to process more data.

2.3.1. File Compression

When storing and manipulating large datasets file compression algorithms allow faster processing and provides more usable space on the disk. Some of the well-known file compression formats that can be used within Hadoop are shown in Table 2.3.

Table 2.3: A summary of compression formats uses on Hadoop

Compression format	**Tool**	**Algorithm**	**Filename extension**	**Splittable?**
DEFLATE (a)	N/A	DEFLATE	*.deflate*	*No*
gzip	*gzip*	DEFLATE	*.gz*	*No*
bzip2	*bzip2*	bzip2	*.bz2*	*Yes*
LZO	*lzop*	LZO	*.lzo*	*No (b)*
LZ4	N/A	LZ4	*.lz4*	*No*
Snappy	N/A	Snappy	*.snappy*	*No*

(a) DEFLATE is a compression algorithm whose standard implementation is zlib. There is no commonly available command-line tool for producing files in DEFLATE format, as gzip is normally used. The deflate filename extension is a Hadoop convention.

(b) However, LZO files are splittable if they have been indexed in a preprocessing step [6].

2.3.2. Codecs

Codec (Compression-Decompression) is the implementation of compression and decompression algorithms that enable faster transmission of data. In Hadoop, a codec is represented by an implementation of the Compression Codec interface (e.g. Gzip Codec encapsulates the compression and decompression algorithm for Gzip). It is the listed that some of the codecs for Hadoop in Table 2.4. [6]

Table 2.4: Compression codes for Hadoop

Compression format	**Hadoop Compression Codec**
DEFLATE	org.apache.hadoop.io.compress.DefaultCodec
gzip	org.apache.hadoop.io.compress.GzipCodec
bzip2	org.apache.hadoop.io.compress.BZip2Codec
LZO	com.hadoop.compression.lzo.LzopCodec
LZ4	org.apache.hadoop.io.compress.Lz4Codec
Snappy	org.apache.hadoop.io.compress.SnappyCodec

2.3.3. Hadoop vs Relational Database Management Systems

E.F. Codd developed the relational database approach in the 1970s. According to his approach relational database supports to ACID (Atomicity, Consistency, Isolation, and Durability) properties. In a relational database system, data is stored in the table which has row and column structure. A relational database management system (RDBMS) is a program that uses a relational model for the data tables and allows to create relational databases as shown in Figure 2.7.

Well known RDBMS depends on SQL or its variants such as SQL, Oracle, MS SQL Server, IBM DB2, My-SQL. The main limitations of RDBMS are scalability and hardware performance on complex queries.

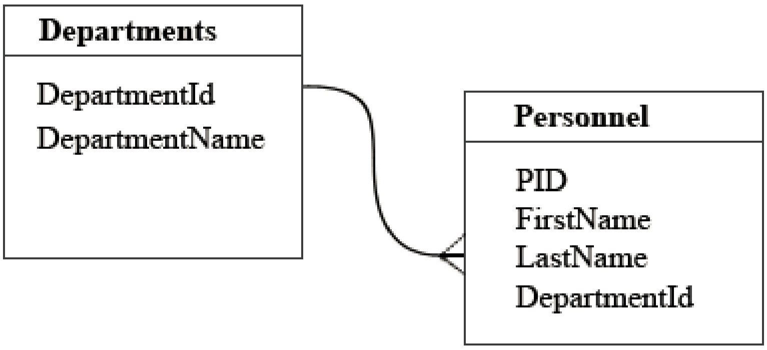


Figure 2.7: A relationship example in a relational database system.

NoSQL (Not only SQL) is a different database architecture approach from traditional SQL architecture in RDBMS. The fundamental difference between SQL and NoSQL approach is the storing data methodology. The data is stored and related to each other column based on NoSQL approach as shown in Figure 2.8. NoSQL databases are more advantageous in terms of scalability than SQL databases. Although the data reaches enormous volume, the performance of data processing does not decrease.

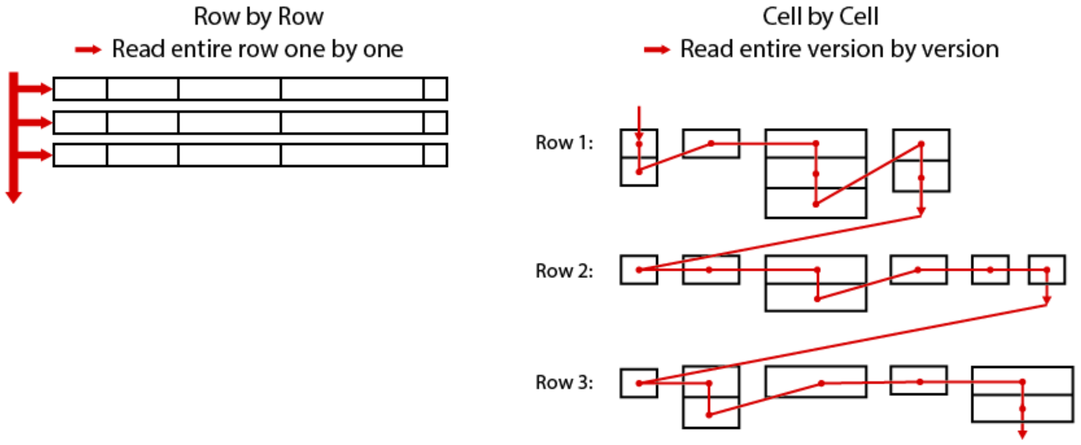


Figure 2.8: The two data store methodologies row by row and cell by cell for SQL and NoSQL architecture respectively.

2.3.4. Hadoop Components

Hadoop had two main components up to the release of version 2.0; Hadoop Distributed File Systems (HDFS) and MapReduce. After Apache Software Foundation (ASF) released the Hadoop 2.0 in 2013, a new component has been added, YARN (Yet Another Resource Negotiator). Today, Hadoop has three main components.

1- Hadoop Distributed File System (HDFS)
2- Hadoop MapReduce
3- YARN

2.3.4.1. HDFS

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware [7]. HDFS is highly efficient for application data and is suitable for applications with large datasets.

A distributed system that uses HDSF probably includes enormous size of server machines inside. HDFS always keeps some resources idle against the possibility of the crash of the machines and uses them to recover quickly.

HDFS structure is more suitable for batch processing instead of interactive usage and the goal is not only the low latency of data access but also the high efficiency of data access.

HDFS aims to achieve high performance through interconnected low-cost machines running on Hadoop in a distributed system. Thus, the application can achieve fast and secure operation at low prices. HDFS supports heterogeneous hardware and software.

HDFS architecture consists of a master (name node) and slaves (data nodes). It stores the data in a block form and blocks in a file and these files are replicated for fault tolerance.

2.3.4.1.1. Architecture

According to the Apache HDFS has a master/slave architecture that consists of a name node (master) and data nodes (slaves). In HDFS architecture clients can read and write data into blocks, the name node manages the metadata (name, replicas) operations and block operations which mapping blocks on data nodes as shown in Figure 2.9.

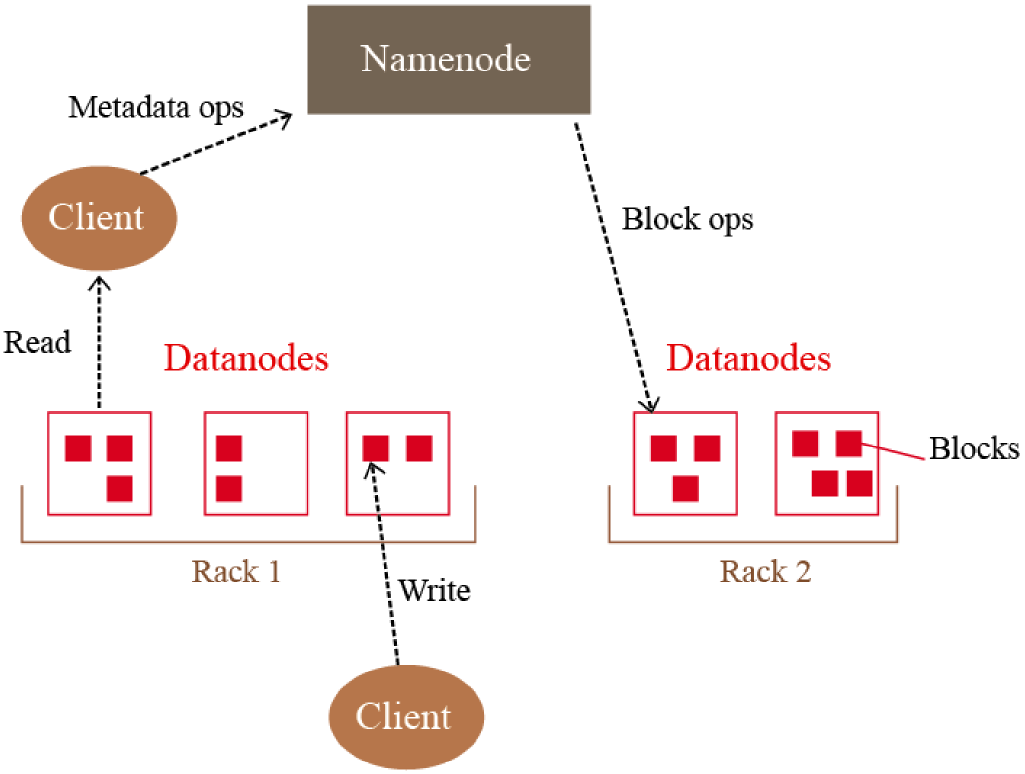


Figure 2.9: HDFS Architecture, block operations and metadata operations between name node and data nodes.

2.3.4.2. Map Reduce

MapReduce is a distributed programming model developed to process vast amounts of data in parallel on thousands of nodes. When the data is processed, two functions known as map and reduce are used. In the Map phase, the data is passed to the worker nodes with the identified task from the master node by separated smaller pieces and as soon as the worker nodes have completed their tasks, the master node returns the result. In the Reduce phase, the result that comes from Map phase is merged and programmer reaches the desired value.

2.3.4.2.1. Map Reduce Algorithm

The programming model of MapReduce is based on key/value pairs in which program written in MapReduce format takes input as a key/value pair and gives output again as a key/value pair. The basic approach is divide and conquer, where we divide a complicated problem into smaller pieces.

There are many implementations of MapReduce such as a count of URL access frequency. According to this implementation, the map function processes logs of web page requests and outputs (URL, 1). The reduce function adds together all values for the same URL and emits a (URL, total count) pair.

Another example is Reverse Web-Link Graph, which the map function outputs (target; source) pairs for each link to a target URL found in a page named source. The reduce function concatenates the list of all source URLs associated with a given target URL and emits the pair: (target; list(source)) [8].

A MapReduce job usually splits the input dataset into independent chunks which are processed by the map tasks in a completely parallel manner [7].

The pseudo-code of the MapReduce program written to execute a word counting problem on a distributed Hadoop system in parallel is given as follows:

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

The map function emits each word plus an associated count of occurrences (just `1' in this simple example). The reduce function sums together all counts emitted for a particular word [8].

2.3.4.3. YARN

After announcing the Hadoop 2.0 version, Apache has developed a component YARN (Yet Another Resource Negotiator) library for resource planning operations that will make Hadoop more powerful. Hadoop enhanced scalability with YARN.

The classic MapReduce framework contains two important components for data processing tasks: Job tracker and task trackers. According to Yahoo!, in this structure, a cluster has a limit and the limit is filled by simultaneous execution of 5,000 nodes and 40,000 tasks.

Apache designed Hadoop to run MapReduce jobs only. With the invention of alternative programming models (such as graph processing provided by Apache Giraph), there is an increasing need of supporting programming paradigms besides MapReduce that could run on the same cluster and share resources in an efficient and fair manner.

YARN is a step added to the MapReduce architecture. One of the reasons for the introduction of this step is that the digital world needs constantly changing and evolving needs such as; scalability, multitenancy, serviceability, locality awareness, secured operations, a variety of programming support.

In the light of these needs, MapReduce's Cluster Manager was transformed into Resource Manager and Job Tracker into Application Master.

After HDFS is installed and formatted to run YARN service on a Hadoop cluster, YARN service can be started with the following commands.

```
# su - yarn
$ cd /opt/yarn/hadoop-2.2.0/sbin
$ ./yarn-daemon.sh start resourcemanager
```

```
starting resourcemanager, logging to /opt/yarn/hadoop-2.2.0/logs/yarn-yarn-resourcemanager-limulus.out
$ ./yarn-daemon.sh start nodemanager
starting nodemanager, logging to /opt/yarn/hadoop-2.2.0/logs/yarn-yarn-nodemanager-limulus.out
```

2.3.5. Relevant Hadoop Technologies

Developed by the Apache Software Foundation for massive data storage and analysis, Hadoop has proven itself in the successful implementation of many companies' projects. Since its first announcement, many Hadoop-related technologies have been developed. Some of which are; PIG provides a platform that facilitates data processing on Hadoop, HIVE which enables you to develop SQL queries to make MapReduce functions easier to develop, MAHOUT which hosts a machine learning library, and SPARK which enables processing on memory without requiring diskette access.

2.3.5.1. PIG

One of the most important components of Hadoop is MapReduce, a method used to analyze data in a large, distributed system. To write a MapReduce program, advanced JAVA programming language knowledge is needed. For this reason, JAVA MapReduce development is tedious, and PIG helps to write MapReduce code faster by reducing this effort.

The Pig used to express the PIG data stream in Latin language and the execution environment required to run the Pig Latin language. There are three different ways to run the Pig program: Script, Grunt, and Embedded.

The following code examples are written in the Pig Latin language:

Pig Latin Example 1:

Query: Get the list of pages visited by users whose age is between 20 and 25 years.

```
users = load ‘users’ as (name, age);
```

```
users_18_to_25 = filter users by age > 20 and age <= 25;

page_views = load ‘pages’ as (user, url);

page_views_u18_to_25 = join users_18_to_25 by name,

page_views by user;
```

Pig Latin Example 2:

Query: Finds the maximum temperature by year

```
records = load ‘input/ncdc/micro-tab/sample.txt’

        AS (year: chararray, temperature: int, quality:int);

filtered_records = FILTER records BY temperature != 9999 AND
        quality IN (0, 1, 4, 5, 9);

grouped_records = GROUP filtered_records BY year;

max_temp = FOREACH grouped_record GENERATE group,

        MAX(filtered_records.temperature);
DUMP max_temp;
```

2.3.5.2. HIVE

HIVE is a data warehouse library developed by Facebook and the Apache Software Foundation that provides creating queries in HDFS using SQL-like statements. HIVE took its current form as Facebook began to develop its daily need to manage the ever-increasing amount of data, and later the Apache Software Foundation community continued to develop.

Hive provides an environment for creating powerful SQL queries for analysts and keeps JAVA coding skills as low as possible.

Below is an example for HIVE. This example shows how to create a table like the RDBMS, load the table into the table, and retrieve it with a SELECT query.

```
CREATE TABLE records (year STRING, temperature INT, quality INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t';

LOAD DATA LOCAL INPATH 'input/ncdc/micro-tab/sample.txt'
OVERWRITE INTO TABLE records;
```

```
hive> SELECT year, MAX(temperature)
> FROM records
> WHERE temperature != 9999 AND quality IN (0, 1, 4, 5, 9)
> GROUP BY year;
1949 111
1950 22
```

2.3.5.3. MAHOUT

Apache Mahout is a distributed linear algebra framework and mathematically expressive Scala DSL designed to let mathematicians, statisticians, and data scientists quickly implement their own algorithms [10]. Mahout includes ready-made libraries that can be created by classification operations to perform machine learning applications.

Apache Mahout focuses on three key areas of machine learning as recommender engines, clustering, and classification. Mahout has three recommender engines are a user-based recommender, item-based recommender, and slope one recommender.

- **User-based recommender:** The recommendation is based on the preferences of similar users.
- **Item-based recommender:** The Item-based recommendation algorithm is based on similar items. It reveals how the items resemble other items.
- **Slope One recommender:** This recommender algorithm is used to map users and items and to evaluate their proximity by using a linearized function.

Apache Mahout is designed for large datasets which separates it from the other alternatives. There are alternative frameworks to Apache Mahout for realizing machine learning classification, evaluation clustering or pattern-mining tasks such as R Community and RapidMiner in the JAVA.

2.3.5.4. SPARK

For all its strengths, MapReduce is fundamentally a batch processing system and is not suitable for interactive analysis [6]. It is hard to say that the result of a query processing on *MapReduce* will return the result within a few seconds. Apache Spark has been developed to meet this demand.

Apache Spark is an open source data processing engine designed to perform in-memory interactive queries at a high-speed. Spark can be used for operations such as stream processing, machine learning applications, and real-time analysis. SPARK is based on an Open Source structure and can integrate with many applications.

Apache Spark is seen as a distributed system programming paradigm that can be used as an alternative to MapReduce, which can run on Hadoop and HDFS. It has an easy to use an run complex processes.

On the speed side, Spark extends the popular MapReduce model to efficiently support more types of computations, including interactive queries and stream processing. Speed is important in processing large datasets, as it means the difference between exploring data interactively and waiting minutes or hours [12].

Initializing Spark and word count application examples codes in Java as shown in Figure 2.10. and Figure 2.11 respectively.

```
import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaSparkContext;

SparkConf conf = new SparkConf().setMaster("local").setAppName("My App");

JavaSparkContext sc = new JavaSparkContext(conf);
```

Figure 2.10: Example of Initializing Spark in Java [12]

```
// Create a Java Spark Context

SparkConf conf = new SparkConf().setAppName("wordCount");

JavaSparkContext sc = new JavaSparkContext(conf);

// Load our input data.
```

```
JavaRDD<String> input = sc.textFile(inputFile);

// Split up into words.

JavaRDD<String> words = input.flatMap(

new FlatMapFunction<String, String>() {

public Iterable<String> call(String x) {

return Arrays.asList(x.split(" "));

}});

// Transform into pairs and count.

JavaPairRDD<String, Integer> counts = words.mapToPair(

new PairFunction<String, String, Integer>(){

public Tuple2<String, Integer> call(String x){

return new Tuple2(x, 1);

}}).reduceByKey(new Function2<Integer, Integer, Integer>(){

public Integer call(Integer x, Integer y){ return x + y;}});

// Save the word count back out to a text file, causing evaluation.

counts.saveAsTextFile(outputFile);
```

Figure 2.11: Example of Word count Java application for Spark [12]

Spark SQL was created to work with structured and semi-structured data. Spark SQL can import data from many different sources and integrates with BI (Business Intelligence) tools such as Tableau, making it possible to use SQL queries. This feature of Spark has made it become widespread. It is shown that an example of Spark SQL imports codes in Java in Figure 2.12.

```
// Import Spark SQL

import org.apache.spark.sql.hive.HiveContext;

// Or if you can't have the hive dependencies

import org.apache.spark.sql.SQLContext;
```

```
// Import the JavaSchemaRDD

import org.apache.spark.sql.SchemaRDD;

import org.apache.spark.sql.Row;
```

Figure 2.12: Spark SQL imports example codes in Java [12]

2.4. Summarization

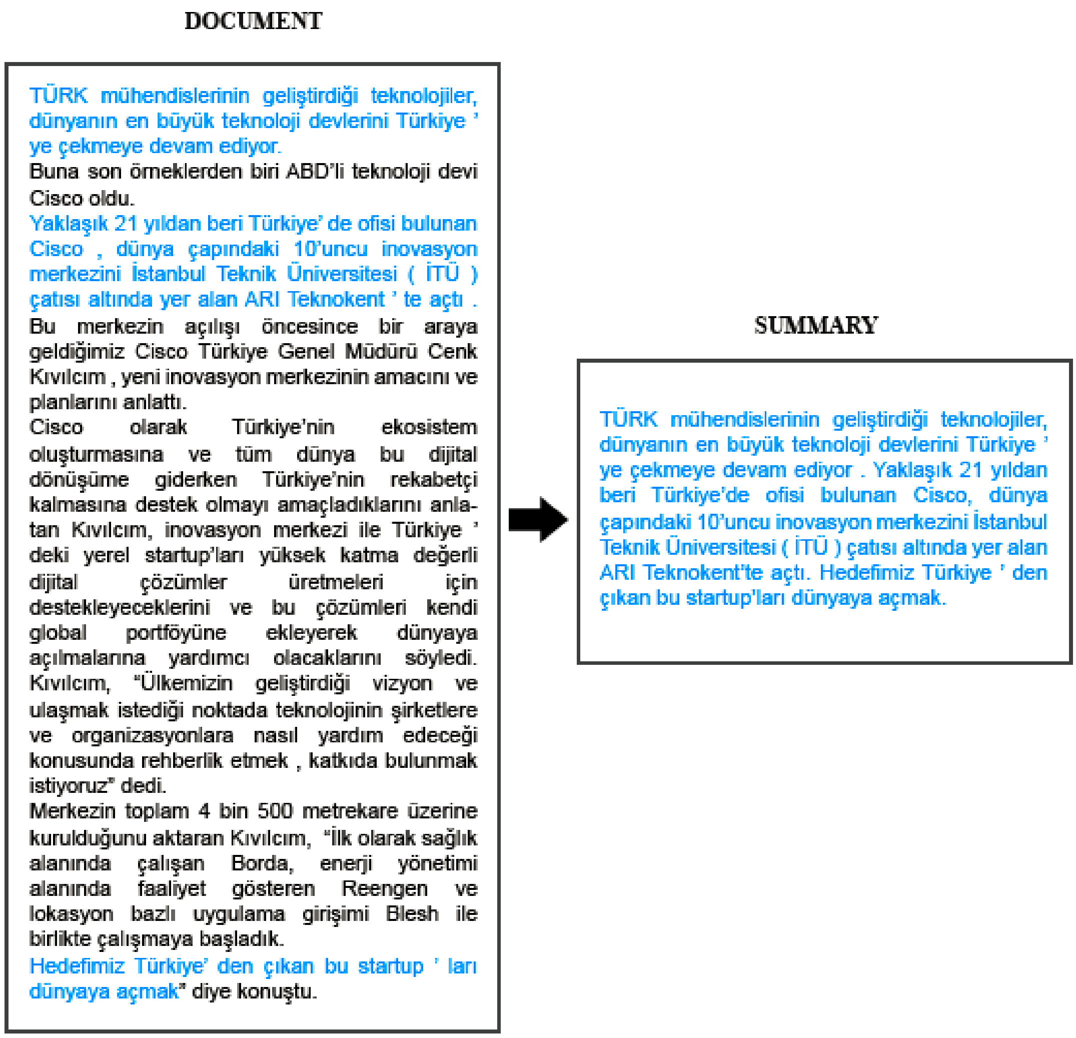
Summarization is the selection of the sentences that are considered important in one or more documents or the process of creating the new sentence(s) or phrases from this context. The automatic text summarization problem, which is addressed in the context of natural language processing studies, has long been one of the most popular topics of researchers working in different languages.

Automatic summarization problems can be categorized in two as single document summarization and multi-document summarization. Multiple document summarization refers to a summary from multiple documents associated with each other, while single document summarization refers to a summary extracted from a single document. In this study, we summarized Turkish news article texts, which is a single document summarization problem.

Another division of summarization problem is given as extractive and abstractive.

In extractive summarization, one extracts important sentences from the text extractive summarization *n* sentences are selected from the main text document and presented as a summary as shown in Figure 2.13

In abstractive summarization, one is constructs new sentence(s) or phrase(s) from the text. The model is based on paraphrasing the content in a couple of sentences from the text. An example abstractive summary is given in Figure 2.14.



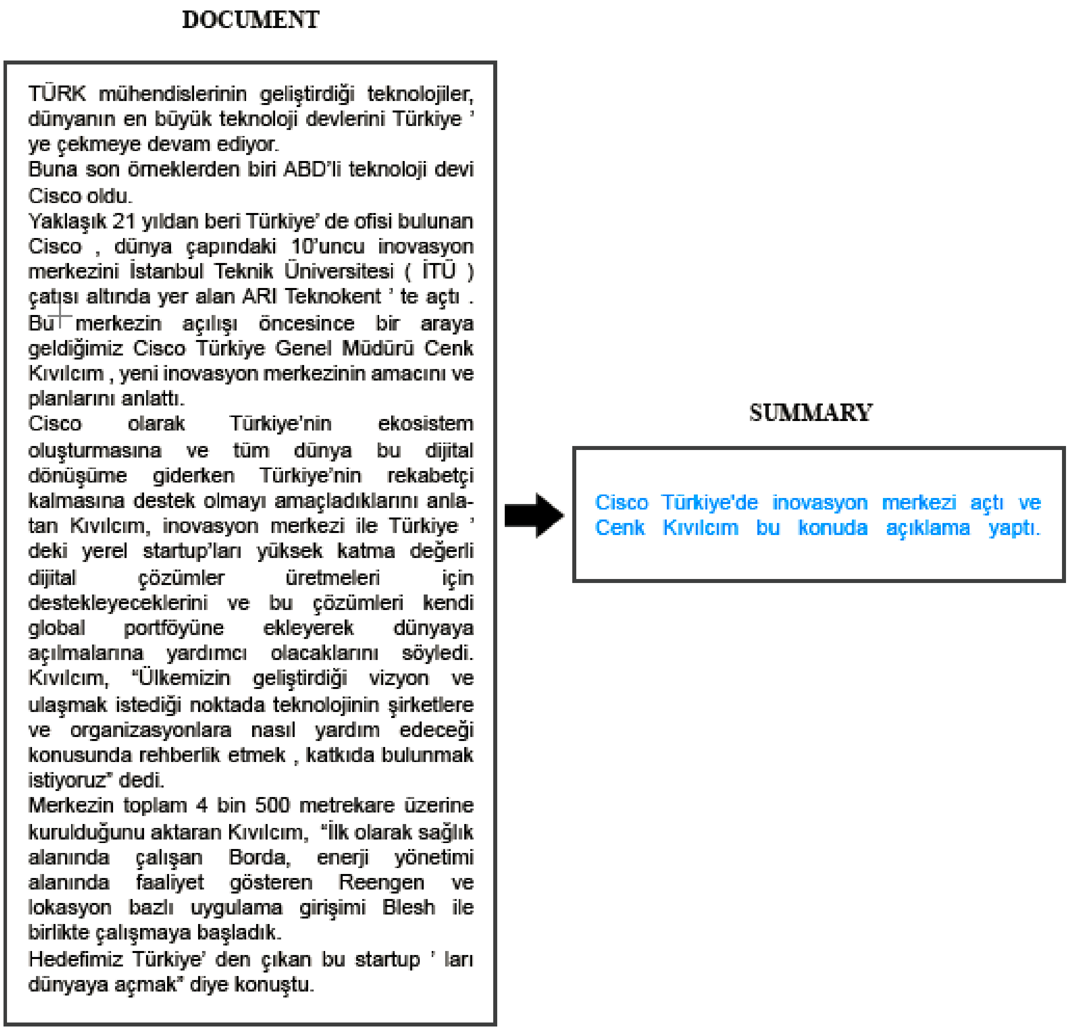
DOCUMENT

TÜRK mühendislerinin geliştirdiği teknolojiler, dünyanın en büyük teknoloji devlerini Türkiye ' ye çekmeye devam ediyor.
Buna son örneklerden biri ABD'li teknoloji devi Cisco oldu.
Yaklaşık 21 yıldan beri Türkiye' de ofisi bulunan Cisco , dünya çapındaki 10'uncu inovasyon merkezini İstanbul Teknik Üniversitesi (İTÜ) çatısı altında yer alan ARI Teknokent ' te açtı .
Bu merkezin açılışı öncesince bir araya geldiğimiz Cisco Türkiye Genel Müdürü Cenk Kıvılcım , yeni inovasyon merkezinin amacını ve planlarını anlattı.
Cisco olarak Türkiye'nin ekosistem oluşturmasına ve tüm dünya bu dijital dönüşüme giderken Türkiye'nin rekabetçi kalmasına destek olmayı amaçladıklarını anlatan Kıvılcım, inovasyon merkezi ile Türkiye ' deki yerel startup'ları yüksek katma değerli dijital çözümler üretmeleri için destekleyeceklerini ve bu çözümleri kendi global portföyüne ekleyerek dünyaya açılmalarına yardımcı olacaklarını söyledi. Kıvılcım, "Ülkemizin geliştirdiği vizyon ve ulaşmak istediği noktada teknolojinin şirketlere ve organizasyonlara nasıl yardım edeceği konusunda rehberlik etmek , katkıda bulunmak istiyoruz" dedi.
Merkezin toplam 4 bin 500 metrekare üzerine kurulduğunu aktaran Kıvılcım, "İlk olarak sağlık alanında çalışan Borda, enerji yönetimi alanında faaliyet gösteren Reengen ve lokasyon bazlı uygulama girişimi Blesh ile birlikte çalışmaya başladık.
Hedefimiz Türkiye' den çıkan bu startup ' ları dünyaya açmak" diye konuştu.

SUMMARY

TÜRK mühendislerinin geliştirdiği teknolojiler, dünyanın en büyük teknoloji devlerini Türkiye ' ye çekmeye devam ediyor . Yaklaşık 21 yıldan beri Türkiye'de ofisi bulunan Cisco, dünya çapındaki 10'uncu inovasyon merkezini İstanbul Teknik Üniversitesi (İTÜ) çatısı altında yer alan ARI Teknokent'te açtı. Hedefimiz Türkiye ' den çıkan bu startup'ları dünyaya açmak.

Figure 2.13: Example of extractive summarization



DOCUMENT

TÜRK mühendislerinin geliştirdiği teknolojiler, dünyanın en büyük teknoloji devlerini Türkiye ' ye çekmeye devam ediyor.
Buna son örneklerden biri ABD'li teknoloji devi Cisco oldu.
Yaklaşık 21 yıldan beri Türkiye' de ofisi bulunan Cisco , dünya çapındaki 10'uncu inovasyon merkezini İstanbul Teknik Üniversitesi (İTÜ) çatısı altında yer alan ARI Teknokent ' te açtı .
Bu merkezin açılışı öncesince bir araya geldiğimiz Cisco Türkiye Genel Müdürü Cenk Kıvılcım , yeni inovasyon merkezinin amacını ve planlarını anlattı.
Cisco olarak Türkiye'nin ekosistem oluşturmasına ve tüm dünya bu dijital dönüşüme giderken Türkiye'nin rekabetçi kalmasına destek olmayı amaçladıklarını anlatan Kıvılcım, inovasyon merkezi ile Türkiye ' deki yerel startup'ları yüksek katma değerli dijital çözümler üretmeleri için destekleyeceklerini ve bu çözümleri kendi global portföyüne ekleyerek dünyaya açılmalarına yardımcı olacaklarını söyledi. Kıvılcım, "Ülkemizin geliştirdiği vizyon ve ulaşmak istediği noktada teknolojinin şirketlere ve organizasyonlara nasıl yardım edeceği konusunda rehberlik etmek , katkıda bulunmak istiyoruz" dedi.
Merkezin toplam 4 bin 500 metrekare üzerine kurulduğunu aktaran Kıvılcım, "İlk olarak sağlık alanında çalışan Borda, enerji yönetimi alanında faaliyet gösteren Reengen ve lokasyon bazlı uygulama girişimi Blesh ile birlikte çalışmaya başladık.
Hedefimiz Türkiye' den çıkan bu startup ' ları dünyaya açmak" diye konuştu.

SUMMARY

Cisco Türkiye'de inovasyon merkezi açtı ve Cenk Kıvılcım bu konuda açıklama yaptı.

Figure 2.14: Example of abstractive summarization

Chapter 3

Experimental Work

3.1. Setup

In this section, the features of the hardware in which process of Big Data stored is executed and the modifications on this hardware will be explained.

3.1.1. Virtual Machine

To model a distributed system on a single machine, we created the virtual machines the on-the-machine basis, and arranged these virtual machines to communicate with each other.

Table 3.1 shows the specifications of the hardware (the server machine at this point) in which store the news text files used for our automated text summarization work and the output summary files of this work.

Windows Server 2012 r2 was installed on the host machine before creating virtual machines. Thus, the necessary ground for the management of the virtual machines to be created using Hyper-V was prepared.

Table 3.1: Server machine specifications

HDD Capacity	1 TB
HDD Speed	7200 RPM
CPU Cache	4 MB Cache
CPU Speed	3,2 GHz
CPU Type	Intel Core i5
Display Card	Intel HD Graphics
Memory Speed	1333 MHz
RAM (System Memory)	8 GB
Ram Type	DDR3

Using Hyper-V, three virtual machines, one master and two slaves, were created and the hardware specifications of these machines were set as shown in Table 3.2.

Table 3.2: Specifications of virtual machines

V-Machine	**IP**	**Gateway**	**DNS**	**RAM**	**HDD**
Master	192.168.1.10	192.168.1.1	255.255.255.0	1 GB	75 GB
Slave 1	192.168.1.11	192.168.1.1	255.255.255.0	2 GB	150 GB
Slave 2	192.168.1.12	192.168.1.1	255.255.255.0	2 GB	150 GB

3.1.2. Distributed Multi-Node Hadoop Cluster

After we created the virtual machines, we installed the Linux Ubuntu 14.04 operating system on the machine designated as the master and we created a Hadoop user to execute Hadoop operations. This user has been granted the necessary permissions. As it is explained about Hadoop being a Java-based software developed by the Apache community and to install Hadoop on the created virtual machines, we installed Java first. After installation of Java program, we displayed *in/usr/lib/jvm/java-8-oracle* directory and defined that directory for the Hadoop system.

3.1.2.1. Hadoop Setup and Adjustment

We downloaded Apache Hadoop version 2.7.2 and extracted it from the folders and moved it to the */opt* directory.

```
tar -zxvf hadoop-2.7.2.tar.gz
```

```
mv hadoop-2.7.2 /opt
```

We have done the necessary configurations on the configuration files (Table 3.3) under placed this directory.

Table 3.3: List of configuration files on path /opt/hadoop-2.7.2/etc/Hadoop

core-site.xml
hadoop-env.sh
hdfs-site.xml
yarn-site.xml
mapred-site.xml

We also edited the masters and slaves files under the same directory. According to these regulations, we entered the hostname of the server named as Name Node in *masters* file and entered specified the hostnames of the machines as Data Node is in the *slaves* file.

We copied these operations on the virtual machine over the Hyper-V tool so that they would not be repeated on other machines, and after we established the SSH connections, they were allowed to communicate with each other in an unencrypted manner.

3.1.2.2. SSH

Secure Shell (SSH) is a protocol developed to allow two machines to communicate with each other in an enhanced manner, such as executing the desired command within another machine to communicate with. There are applications like Putty that use this protocol. The SSH protocol includes two applications inside, Secure Shell Client and Secure File Transfer Client.

After we copied the virtual machines for our project, SSH connections were created to communicate securely with each other. This was done by downloading OpenSSH and then installing OpenSSH -server and OpenSSH -client.

sudo apt-get install openssh-server

sudo apt-get install openssh-client

We created private and public keys for each machine that would communicate with each other.*ssh-keygen -t rsa*

We configured each machine to recognize other machines that would communicate (known-hosts).

ssh-copy-id hadoop@master

ssh-copy-id hadoop@slave1

ssh-copy-id hadoop@slave2

In the meantime, for those machines that recognize each other, we defined a preliminary user and password.

3.1.2.3. HDFS

As already mentioned, HDFS is a file system created for Hadoop. This file system needs to be formatted to make it available. We managed Hadoop cluster through Name Node so while we are in Name Node,

hdfs namenode–format

command formats the file system. Then Hadoop is started;

start-dfs.sh

The *jps* command can be used to see which Hadoop process is taking place.

3.2. Dataset

In order to create the Turkish text dataset to be processed, we preferred to collect the texts which were passed by an editor and which had no language information mistakes. Turkish news articles are an ideal dataset for this task.

3.2.1. News Articles Collection

The news articles data required for our work was obtained through the API (Application Programming Interface) presented by Hürriyet Newspaper, which opened Turkish news feeds to software developers. [14] We downloaded a total of 100,000 news articles for this study.

3.2.2. Hurriyet Search API

For those who want to use the service provided by Hürriyet, it is necessary to create an API Key first. With this API key and the keyword to be searched, a web service query was created for Hurriyet's Search API, and the JSON-formatted response was parsed and prepared for preprocessing before the summarization operation.

3.2.3. Preprocessing of Data

Parsed data was separated into sentences and paragraphs with the help of JSON tags. We added each sentence as a line and we added a blank line for the line preceding the paragraph. We removed the unnecessary points and words in the news from the text. The data passed from the pre-summarization process were filed and recorded separately as the date of the notice and the ID information and the title, spot, and middle text.

3.3. Batch Data Input

The data from the news source is stored in files. We thought of this data as Big Data with the idea that the file size could increase and that the operations could be related to the documents in the future. We stored both the files before and after the summarization process in the Hadoop system that we created beforehand. The application *Putty* was used for this and data was written over HDFS to be kept in the predefined *Data Node* machines. This was exploited by the Apache-supplied script. A total of 100,000 news documents were stored on the distributed system.

3.4. Summarization Methodology

Suffixes are added at the end of the words in Turkish. That is to say, the words are eventually enriched by one or more additions to become different words. This agglutinative Turkish brings various difficulties in the implementation of natural language processing processes. One of these difficulties is encountered with the application of a process such as finding the meaning of the word.

In order to apply the *proximity matrix* method we used in our Turkish text summarization study, we reached information on the morphological level of the words. This process took an important role in determining the action of the text and was used in the selection of the important phrases in the text.

In order to perform the automatic summarization process, probable sentences were firstly selected and morphological analysis method was applied to each word in order to find the root words passing through these sentences.

Subsequently, the root words were evaluated according to their closeness to each other and a word sentence matrix was formed. After the reduction made to this matrix, summary sentences were obtained.

In this section, morphological analysis, scoring processing operations and the phase of sentence selection will be explained.

3.4.1. Morphological Analysis

The method of morphological analysis, which is often used in natural language processing, is a set of procedures that allow the structure of words to be extracted. Morphological analysis is a method in which the distinction of roots and suffixes is made and changes the form of the word to a verb, adjective or noun are determined according to the suffixes to the root word.

In this study, it is benefited from the work whom Onur Görgün and Olcay T. Yildiz did for Turkish morphological analysis named “A Novel Approach to Morphological Disambiguation for Turkish [1]”.

The probable roots of each word were estimated by the Final State Machine process. According to this method, each word was compared with a root database and the root word was identified.

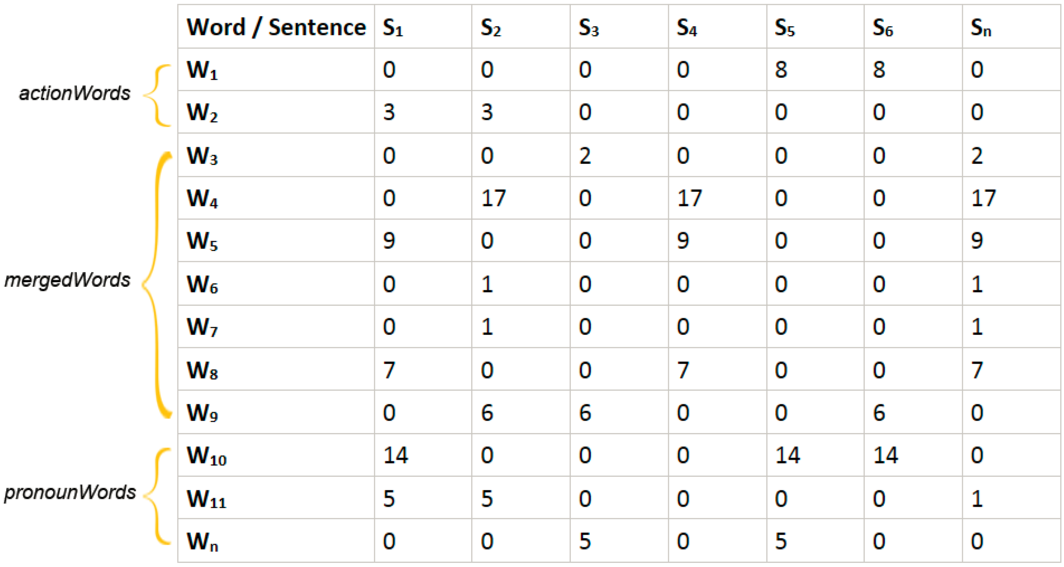
With the transition process, changes in species of the word were determined by the additions of the suffix to the roots of words. In Turkish words attachments are added after the root. Every root can gain new meaning with suffix or suffixes. To determine this, each root is evaluated according to all word types (such as name, verb or adjective) and an additional guess is made. With this method, it is possible to determine which word type belongs to by looking the first and last value of the word. The first and last values of the word are evaluated together while the action of the document is found. The first and last values of *action words* must be verbs.

3.4.2. Scoring Process

Selected sentences were scored to identify important sentences. The methodology used for these scoring procedures is as follows.

- Action words, which are the biggest help to find the meaning of the document in general, were determined. Accordingly, the words that may be the *action* that is the first and last form is the verb of the document are listed and they take a point value as much as the number passed on the document.

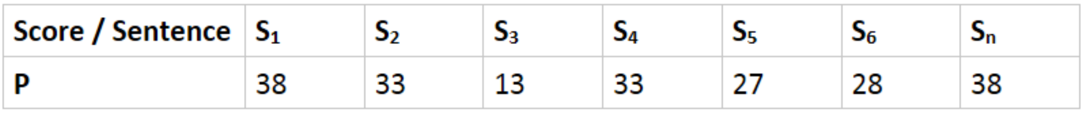
- It has been observed that long sentences adversely affect the summarization performance of the text. Therefore they are deleted.
- A hash map has been created to detect frequently used words. Accordingly, the meaningful roots of the words were rated according to the number of exists in the document.
- The conjunctions in the sentence were removed from the list because they did not express a value when selecting meaningful sentences.
- A list was created for the special names on the document, and each name is rated according to its total number in the document.
- The first sentence of the document and head sentences of the paragraphs were determined and given +1 points to these sentences.
- A threshold was set for all these word lists and a two-dimensional proximity matrix was formed that contained the words on this threshold and the previously selected sentences (Figure 2.12). Sentences in this word-sentence matrix were scored according to the state of having the top words that most used in the document.



	Word / Sentence	S_1	S_2	S_3	S_4	S_5	S_6	S_n
actionWords	W_1	0	0	0	0	8	8	0
	W_2	3	3	0	0	0	0	0
mergedWords	W_3	0	0	2	0	0	0	2
	W_4	0	17	0	17	0	0	17
	W_5	9	0	0	9	0	0	9
	W_6	0	1	0	0	0	0	1
	W_7	0	1	0	0	0	0	1
	W_8	7	0	0	7	0	0	7
	W_9	0	6	6	0	0	6	0
pronounWords	W_{10}	14	0	0	0	14	14	0
	W_{11}	5	5	0	0	0	0	1
	W_n	0	0	5	0	5	0	0

Figure 3.1: Example of proximity matrix shows that how many every important word included in selected sentences.

The reduced matrices were created with redundant records in proximity matrix. (Figure 2.13).



Score / Sentence	S_1	S_2	S_3	S_4	S_5	S_6	S_n
P	38	33	13	33	27	28	38

Figure 3.2: Example of the reduced matrix shows that total scored for each sentence.

3.4.3. Summarization Phase

After applying the scoring methodology to all the sentences, the position of the index of the first three scales in descending order was taken. We extract the corresponding sentences to these selected indexes from the document and created a summary file.

The summary files created for each document were associated with the name and date of the news files and recorded on HDFS.

Chapter 4

Analysis

Summarization is person-dependent, so a text that someone sees as a summary may not mean the same thing for another person. In this study, we developed a methodology taking into consideration the generally preferred methods for extracting a summary text.

In this section, you will be given information about how the performance of the summary text obtained as a result of the study is measured.

4.1. Test

In this study, the scoring method of the sentences was applied in order to obtain the summary inference. All sentences were passed through a scoring step shown in Table 4.1 after being passed through a certain length control.

Table 4.1: Criteria considered for sentence rating.

The sentence containing the action of the text.
The sentence that included the most used special name.
The sentence that included the date.
The sentence containing the most used words.
The sentence that head of text or paragraph.

After these scoring steps we obtained summary files then we compared these files with the results comes from testers. The testers consist of 20 different people in characteristics of 12 women, 8 men, 7 student and 13 employees.

We shared 50 pieces of documents with the testers. Unaware of each other, we asked from the tester to select three sentences that were important from these documents and record them in a file. These test documents consist of Turkish news articles and average length all of these articles approximately 20 sentences.

We have three test scenarios these are comparison similarity between program's summaries with testers' summaries, the similarity between testers' summaries themselves, and similarity between testers' summaries and random summaries obtained by extracting the sentence from text randomly from every news articles.

The results obtained figured in the result section

4.2. Result

When we look at the results (Table 4.2) we see that the similarity performance measured among testers has the highest similarity (41.5 %) compared to other measures. The reason for this may be that the testers use different summarization methods for each news text. When our program applies a methodology that is the same as all texts, the tester may be applying different methodologies to different texts. This can affect the emergence of high similarity rate.

Similarity ratios of the summaries obtained from the program and the summaries taken from the tester are found to have the second highest similarity rate (36.5 %). As we have said before, summarization is a person-dependent process and may vary from person to person. In the program we created in the study, we compiled certain rules that people were aware or not of when summarizing, and created a method. When we applied this method, we see that the result of the summarization of the program is close to the similarity rate between people's themselves - 5 percent difference.

The lowest similarity rate is the result of, comparing the summary that randomly created summaries with the summaries from the testers (11 %). This is because the summary text generated from the randomly selected sentences is not based on any methodology. Therefore, when we compare the random result with the result that we obtained from the program, we see that there is a similarity difference of 25.5 % between them. This result shows us that it is important to use a methodology in the summarization process.

Another general observation is that the summation rates in the long texts are less similar than the summation rates in the short texts. This is because the number of sentences that can be extracted from long texts is much larger and therefore the possibility of selecting alternative sentences increases.

Table 4.2: The result values of three different similarity ratio calculation operations. The results were obtained by comparing the similarity of all the reports with each other. Accordingly, the similarity rate of the summaries of the program to the summaries of the testers is calculated as 36.5% on average. The similarities of the summaries extracted from between the testers themselves were 41.5%. The similarity rate of the summaries obtained from the randomly selected sentences and the summaries obtained from the testers' was calculated as 11%.

News article text no	Similarity rate between program and testers results (%)	Similarity rate between testers results themselves (%)	Similarity rate between testers and random results (%)
1	42	43,5	9
2	30	52,5	10
3	37	35	9
4	27	29,33	6
5	52	30,17	9
6	37	29,5	20
7	40	29,83	1
8	40	30,17	0
9	32	32,33	3
10	38	33,83	1
11	45	37,83	33
12	42	31,83	19
13	25	39,67	1
14	37	26,67	18
15	35	27,33	21
16	45	54,5	22
17	37	39,17	3
18	40	47,67	3
19	34	50,5	13

20	27	50,17	7
21	35	52,5	17
22	29	39,5	2
23	32	45,5	1
24	34	31,17	7
25	29	42,83	6
26	40	31,67	0
27	35	50,67	18
28	35	24,83	3
29	39	51,83	10
30	37	53,83	21
31	34	51,33	26
32	42	38,5	3
33	37	51,33	2
34	29	36,67	20
35	30	23,83	9
36	35	29,17	8
37	39	37,83	19
38	32	48	20
39	32	33,33	22
40	47	51,33	14
41	35	49,83	1
42	49	51,33	7
43	27	41	14
44	27	55,17	8
45	37	47,67	12
46	45	58,83	26
47	35	56,17	6
48	25	44,33	10
49	22	35,33	0
50	35	59,17	0
Average	**35,6**	**41,5**	**11**

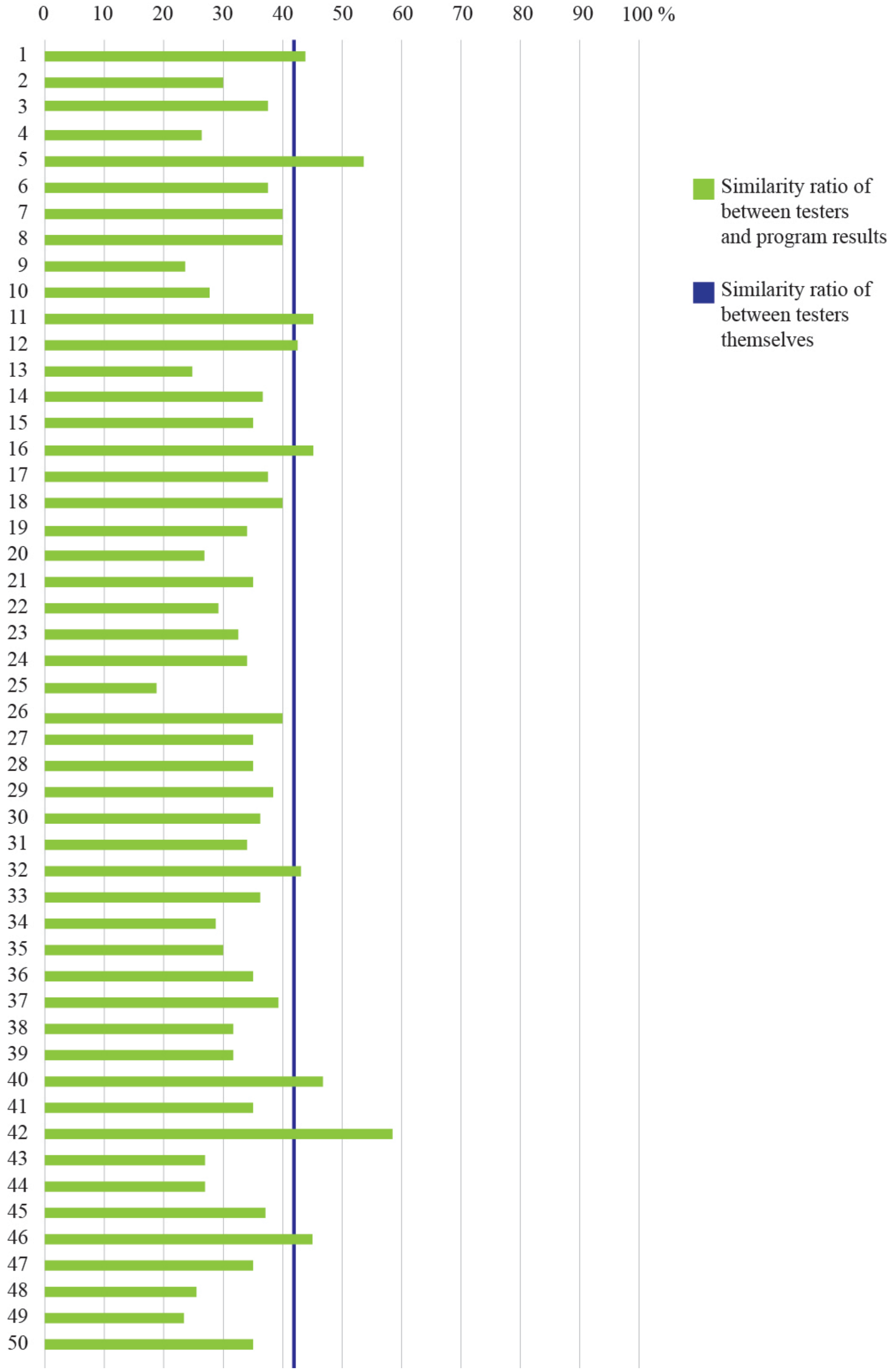


Figure 4.1: Similarity performance rates comparison of program summaries and 20 testers' summaries for each news article text. We have done this performance measurement as follows; the program extracted a summary file for each news, while the testers produced a summary file for each news article. For each news, we compared the summary of each tester's output to the program summary and we obtained 20 comparative results for each news. Each bar in the figure shows the value obtained from the average of these 20 results. The detailed description is shown in Figure 4.4.

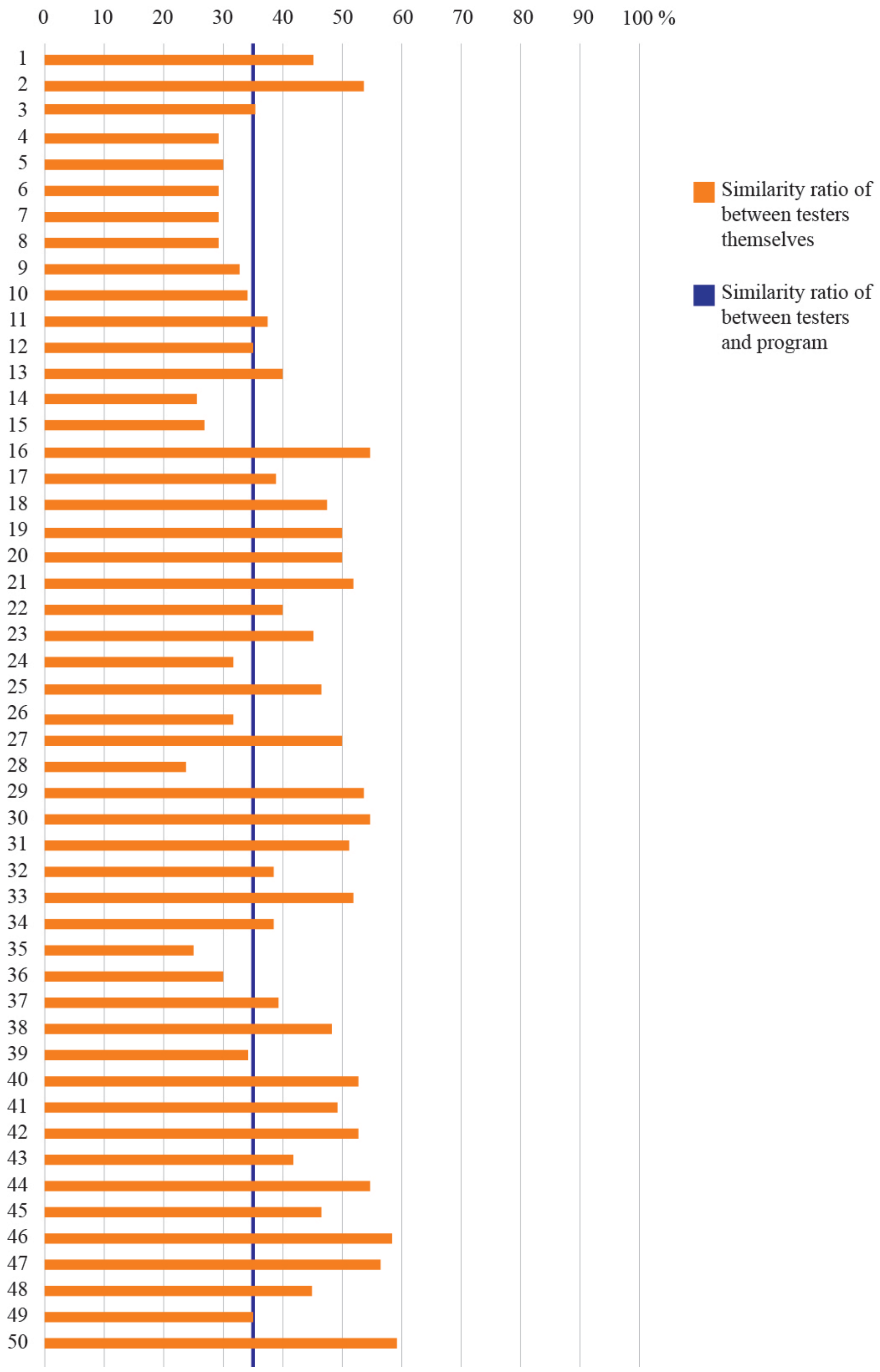


Figure 4.2: Similarity performances rates comparison of 20 testers' summaries themselves for each news article text. We have done this performance measurement as follows; testers made a summary for every news text and we got 20 summary files for each news in total. We rated these 20 files according to their similarity to each other and we calculated 20 points in this way. Each bar in the figure shows the average of these 20 points. The detailed description is shown in Figure 4.5.

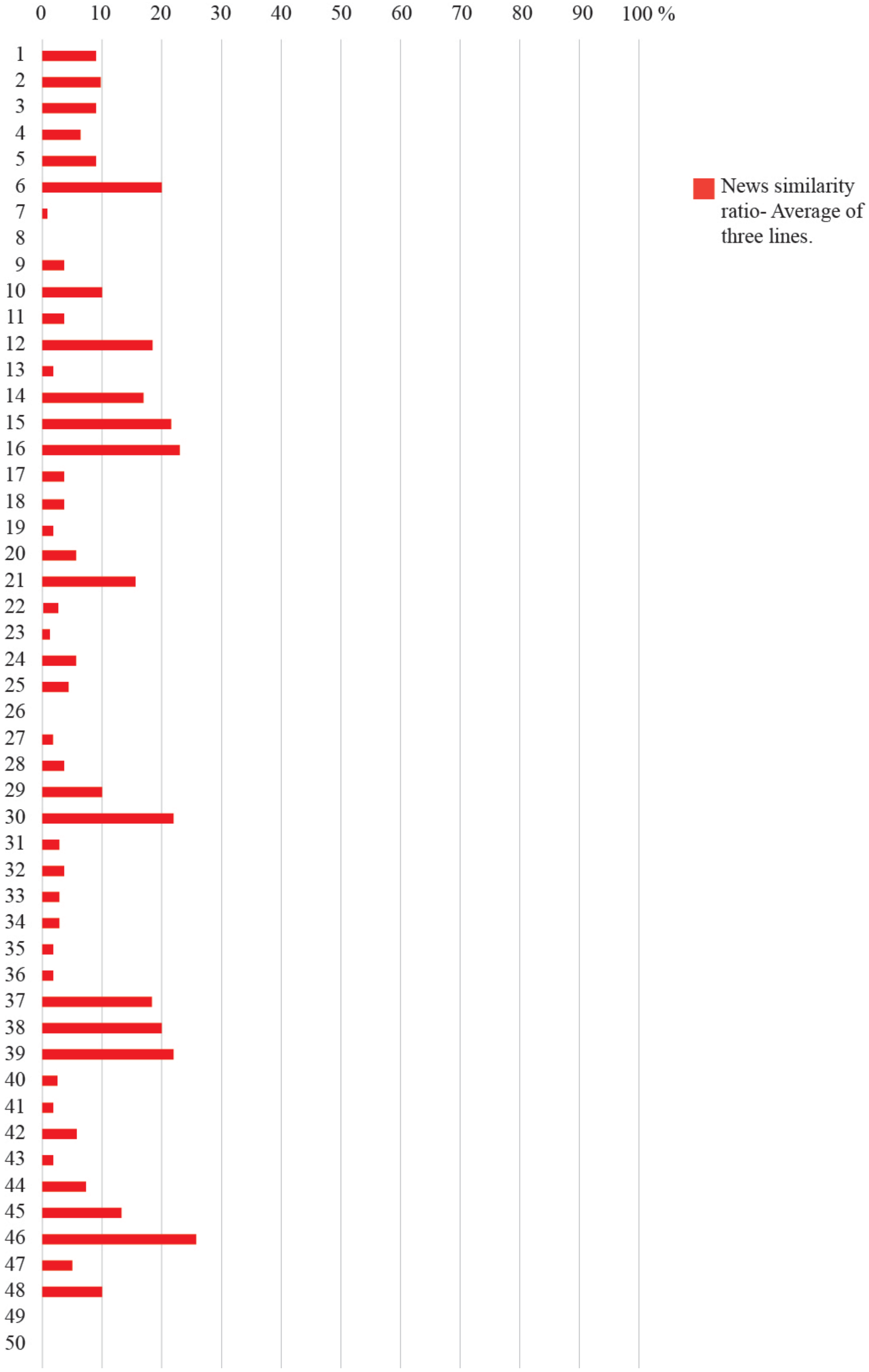


Figure 4.3: Similarity performance rates between random summaries and testers' summaries. We have done this performance measurement as follows; with a program, we created a summary file of 3 sentences randomly selected from each news text then we compared each outcome with the summaries of each tester. Each bar in the figure shows the average of these results. The detailed description is shown in Figure 4.6.

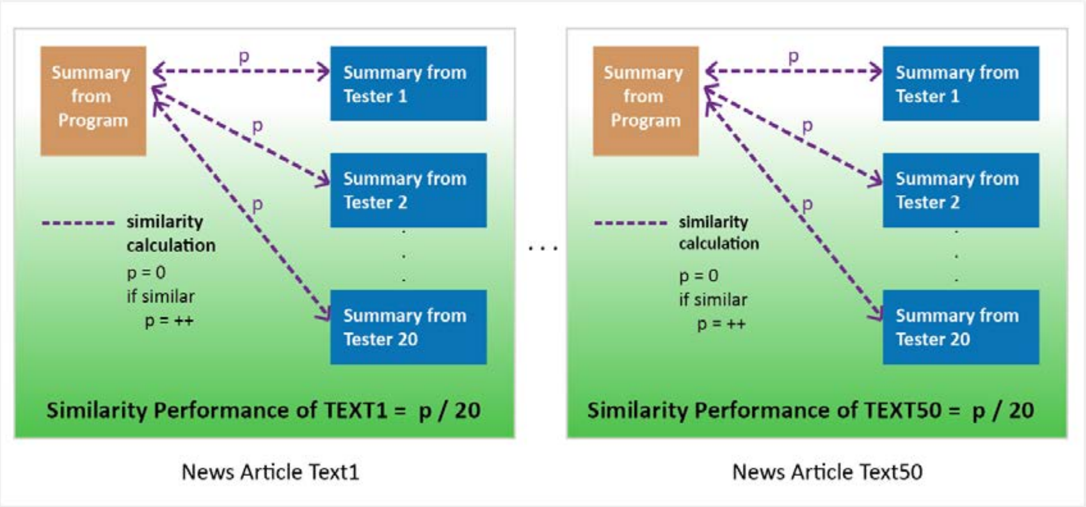


Figure 4.4: The detailed description of similarity comparison method between program and testers'.

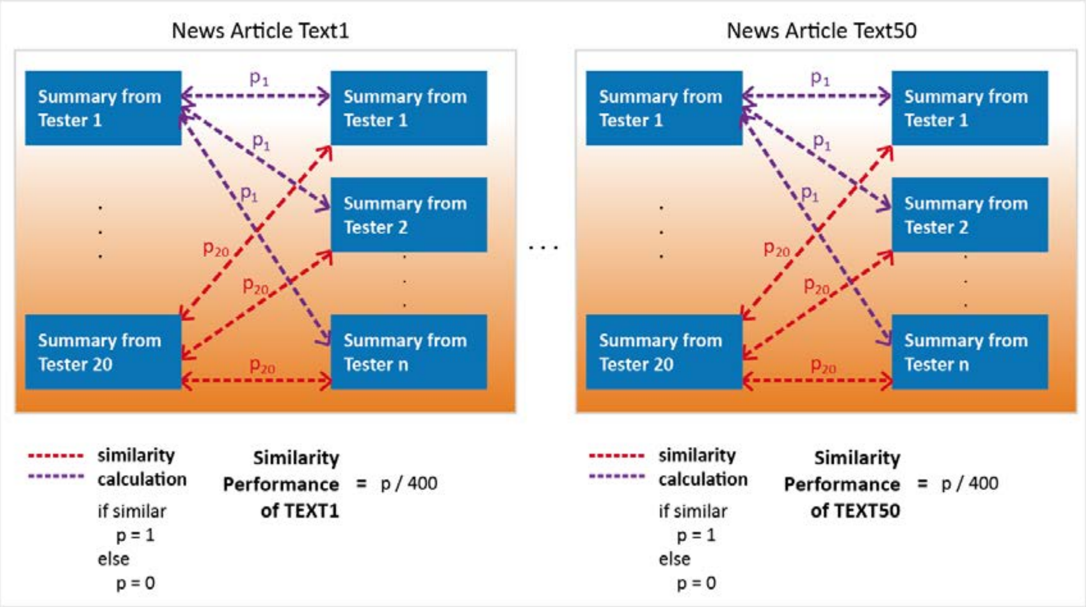


Figure 4.5: The detailed description of similarity comparison method between testers' result themselves.

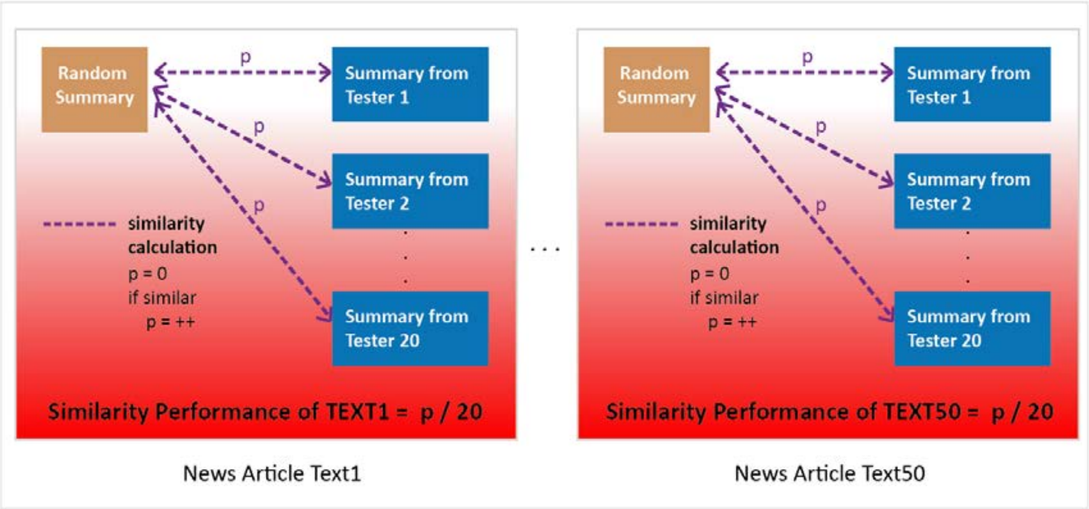


Figure 4.6: The detailed description of similarity comparison method between summary created randomly and testers' summaries.

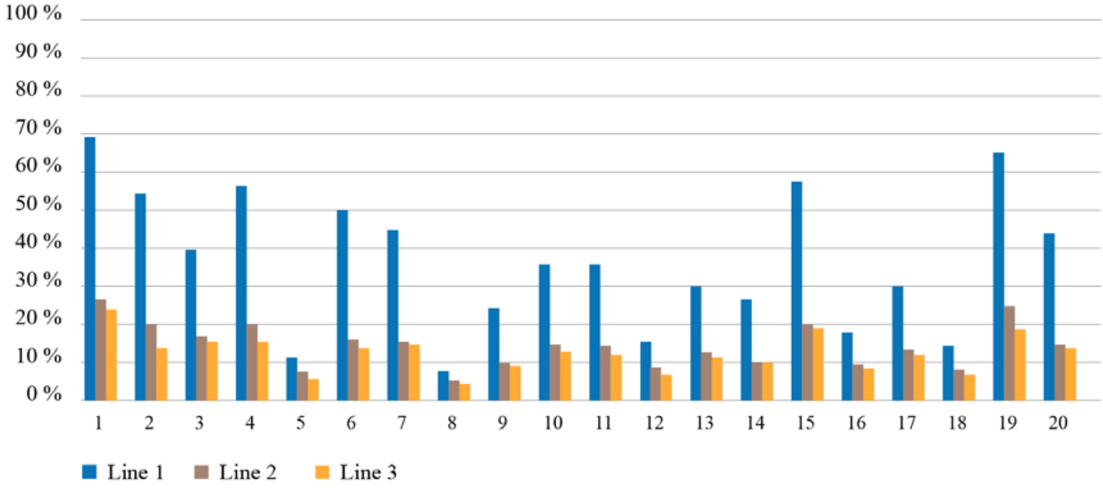


Figure 4.7: Similarity performance rates of each tester according to the average of all 50 news article texts – for each of three summary sentence. We have done this performance measurement as follows; the summary text of the study consists of 3 sentences. That's why we asked for testers to extract a three-sentence summary of the text given to them. We also calculated sentence to sentence similarity ratios while each summary of 20 testers' is compared with each other. These rates are calculated for 50 different news and averages were taken. The blue bar in the figure shows the similarity of the first sentence, the brown bar shows the second sentence and the orange bar shows the third sentence. As you can see in the figure, the sentence with the highest similarity is in the first sentence. It is understood that each user applies the similar method to choose the first sentence.

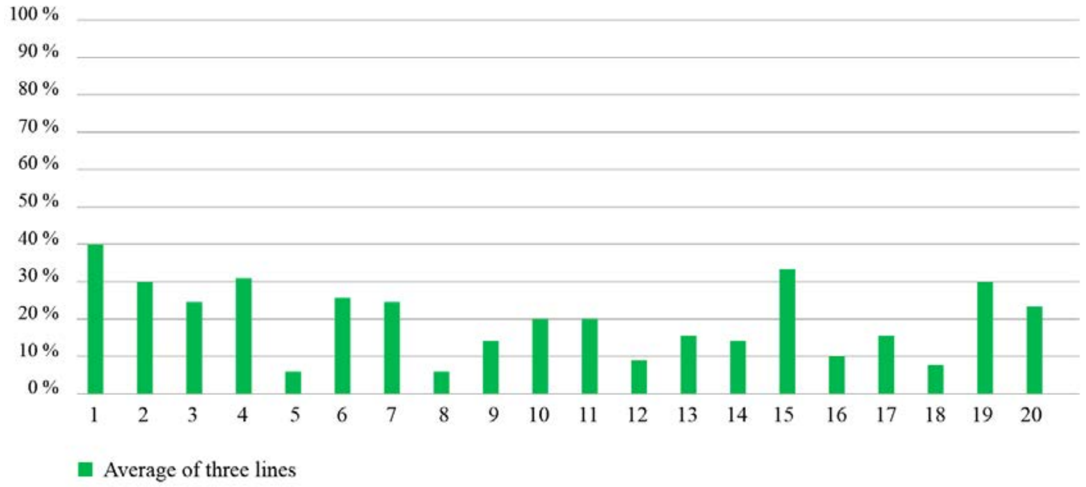


Figure 4.8: Similarity performance rates of each tester according to the average of all 50 news article texts – for each of three summary sentence. We have done this performance measurement as follows; we compared the 20 tester's summaries for each news based on similarities to each other and we got a total of 50 results for each tester. Each bar in the figure shows the average of this 50 results obtained.

Chapter 5

Discussion

Is it really true that every large datasets are Big Data? Actually, Big Data is not a term that refers just lots of data. It refers that large of datasets which cannot handle with traditional database management systems. Another possible explanation is that Big Data is an ecosystem that contains a huge amount of data and new storage and analyses systems inside.

Why Hadoop? It is aimed to store the dataset used in this study and the files in which the results of the study are kept in an expandable system. To do this, Hadoop was preferred because Hadoop is a scale-based structure that has the HDFS file system and allows you to add new nodes without losing performance to the cluster.

5.1. Future Work

A proximity matrix is generated and a reduced matrix derived from this matrix in this study. Our next goal in a study is to derive a summary sentence from a combination of related sentences using these matrices. According to this, a path will be drawn between the elements of the sentences with meaningful closeness and it is aimed to construct an algorithm which finds the elements such as the subject or predicate of the sentence in the direction of this path.

This work establishes the single document summarization process. Our next study aims to create an algorithm that scans all the text documents on our Hadoop cluster and finds each other's related documents and performs their automatic summarization (multi-document summarization).

Chapter 6

Conclusion

It is introduced in this study that how data grow in time and shaped new forms named Big Data. Due to the processing of Big Data with machine learning methods and the efficient results, artificial intelligence-based applications have increased tremendously. Studies on Big Data and related artificial intelligence has started a new era in the industry.

In this study, which describes a natural language processing (NLP) subdivision of artificial intelligence, and an application of automatic text summarization on Turkish, which is a subdivision of NLP, an automatic summarization method of sentence extraction was applied. As a dataset, Hurriyet Search API which Hurriyet Newspaper opened the news articles to software developers was used. Using this API, a high amount of news text was downloaded and stored on a Hadoop cluster, under certain processing. These texts were then used to pass through the summarization process.

In this study, each word was subjected to morphological analysis in order to determine the closeness of the words. With the help of the morphological analysis method, the word types were determined according to the roots and the suffixes they received. The determination of the types of the words played an important role in the selection of the important words in the text.

We measured the study period of the summarization program included in this study. In order to compare the results of the program with the results of the testers, the program was firstly run -via a personal computer- for 50 news text and the duration of the study was approximately 10 minutes. When we run the program with 100,000 news text entries on the Hadoop multi-node cluster established in this study, it is measured that the working time will be approximately 3 days. It is seen that the duration of this study will decrease in proportion to the larger cluster structure.

Finally, we obtained the results from the study were compared with the results from the persons, a performance measurement of approximately 37 % was provided and similarity performance of results between testers themselves is calculated approximately 42 % and we measured the similarity performance between random summaries and testers’ summaries is approximately 11 %.

References

[1] O. Görgün and O. T. Yildiz, “A novel approach to morphological disambiguation for turkish," in Computer and Information Sciences II. Springer, 2011, pp. 77-83.
[Online]. Available: http://haydut.isikun.edu.tr/cv/bildiri/ISCIS-2011-1.pdf

[2] Nallapati et al. “SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents” in The Thirty-First AAAI Conference on Artificial Intelligence (AAAI-2017)

[3] Rada Mihalcea and Paul Tarau “TextRank: Bringing Order into Texts” in Conference on Empirical Methods in Natural Language Processing (EMNLP), 2004, Barcelona, Spain

[4] Ebru UZUNDERE, Elda DEDJA, Banu DİRİ, M.Fatih AMASYALI “Türkçe Haber Metinleri İçin Otomatik Özetleme” in Innovations and Applications in Intelligent Systems Symposium, 2008, Istanbul, Turkey

[5] Sara Landset, Taghi M. Khoshgoftaar, Aaron N. Richter and Tawfiq Hasanin “A survey of open source tools for machine learning with big data in the Hadoop ecosystem”
Journal of Big Data, Published: 5 November 2015

[6] Hadoop: The Definitive Guide – Storage and Analysis at Internet Scale A Book from Tom White – 4.Edition, pp. 6,100,101

[7] https://hadoop.apache.org/
Access Time: 10.05.2018
Page: HDFS Architecture Guide / Introduction

Access Time: 12.05.2018
Page: MapReduce Tutorial / Overview

[8] Jeffrey Dean and Sanjay Ghemawat, Google, Inc.
"MapReduce: Simplified Data Processing on Large Clusters." In OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA (2004), pp. 137-150

[10] https://mahout.apache.org/
Access Time: 10.05.2018
Page: Mahout / Introduction

[12] Learning Spark: Lightning-Fast Data Analysis
A Book from Holden Karau, Andy Konwinski, Patrick Wendell & Matei Zaharia, pp. 1,18,19,165

[14] https://developers.hurriyet.com.tr/docs/versions/1.0/resources/search
Access Time: 10.05.2018
Page: Hurriyet Developers API