# Software Requirements Specification

**Prepared by Default**

**for the project Süzgeç**

**(Turkish Text Summarizer with Deep Learning)**

Dr. Ayşenur Birtürk - *Supervisor*

Itır Önal - *Project Assistant*

**Team Members**

Abdullah Göktuğ Mert - 1881390

Baran Barış Kıvılcım - 1881325

Enes Uğur Şekerci - 1942689

Yağız Arkayın - 1880988

*METU - Department of Computer Engineering*

*CENG 491 Senior Design Project I*

*Fall 2015-2016*

## 1. Introduction

This document contains the system requirements for Süzgeç Project. It is prepared by the team Default.

This specification document includes descriptions of the functions and the specifications of the project Süzgeç.

In this section(Section 1), a review of the entire document is provided. The reader would get familiarized with the contents before the further details are described.

### 1.1 Problem Definition

People need to learn much from texts. But they tend to want to spend less time while doing this. We aim to solve this problem by supplying them the summaries of the text from which they want to gain information. Our goals are that

- These summaries will be as important as possible in the aspect of the texts' intention
- The user will be eligible to select summary length
- Supplying the user a smooth and clear interface
- Configuring a fast replying server system

### 1.2 System Overview

The product is mainly a text summarizing Google Chrome extension using Deep Learning concepts. The main purpose is to provide reliable summaries of web pages or uploaded files depends on the user's choice. The unnecessary sentences will be discarded to obtain the most important sentences.

The product includes the following components:

- Text Parser: It will divide the texts into paragraphs, sentences and words.
- Feature Vector Creator: This component will calculate and get the feature representations of sentences.
- Autoencoder: The root part of the Deep Learning. Autoencoder offers a compressed representation of a given sentence.
- Classifier: The classifier determines if a sentence is a summary sentence or not.

An explanation in detail is in the section 4.1.

### 1.3 Definitions, Acronyms, and Abbreviations

The document contains words and abbreviations related to computer science. The terms and abbreviations are the following ones:

*Server:* A program that awaits and fulfills requests from client programs in the same or other computers.

*Deep Learning: A machine learning method that stimulates the neural network in human brain.*

*C:* A high level general purpose programming language.
*Java:* An object oriented programming languages that compose some of the tools we use in this project.
*Python:* One of the most commonly used programming languages.
*Matlab:* An environment and its embedded programming language mostly used for mathematical calculations or machine learning purposes.
*Autoencoder:* The core part of the neural network system that is used in deep learning.
*Classifier:* An algorithm used in Statistics and Machine Learning areas to divide the data in two or more classes.
*Word2Vec:* A matrix representation of words and their meaning similarities.
*Kernel:* The lowest level of easily replaceable software that interfaces with the hardware in your computer.
*IEEE:* Institute of Electrical and Electronics Engineers
*Zemberek:* A Java NLP API for Turkish language developed by TUBITAK.
*TRNLTK:* A Python NLP API for Turkish language.
*NLP:* Natural Language Processing
*API:* Application Program Interface
*Tokenizer:* A component that breaks up a sequence of string into pieces such as words, phrases, symbols and other elements called tokens.

### 1.4 Assumptions and Dependencies

- The server must support Linux Kernel
- Our methodology of combining two probabilities (explained in detail in the following sections) is not desirable, one of these algorithms serving probability can be discarded.
- The designed algorithm is for Turkish language and the text the application parses must be written in this language.
- The interface of resulting system will be easy to use and accessible without a time or location constraint.
- The user must have Google Chrome as Web Browser.
- The application requires a stable network connection.
- The user must have a google account to install and use the extension.
- If one of component does not work on Linux the programming language of it can be replaced with another one.

## 2. Overall Description

This section is about the requirements, constraints and the interfaces included in the project. A map of functions are also supplied.

The document follows the IEEE standards, yet some of the sections are discarded as they are not compatible for this project.

## 2.1 Product Functions

Major functions of the product and brief descriptions of these functions can be found in this section. Also detailed diagrams and descriptions can be found in subsections of this section.

| ID | Function | Description |
|---|---|---|
| 1 | Summarize Web Page | Getting summary of a webpage |
| 2 | Summarize File | Getting summary with uploading a file |
| 3 | Summary Setting | Setting the length of the summary |
| 4 | Train System | Training the system’s machine learning part for better result |

### 2.1.1 Use-Case Model Survey

This section includes use case diagrams and their detailed descriptions of the functions that mentioned in section 2.1.

- **Summarize Web Page**

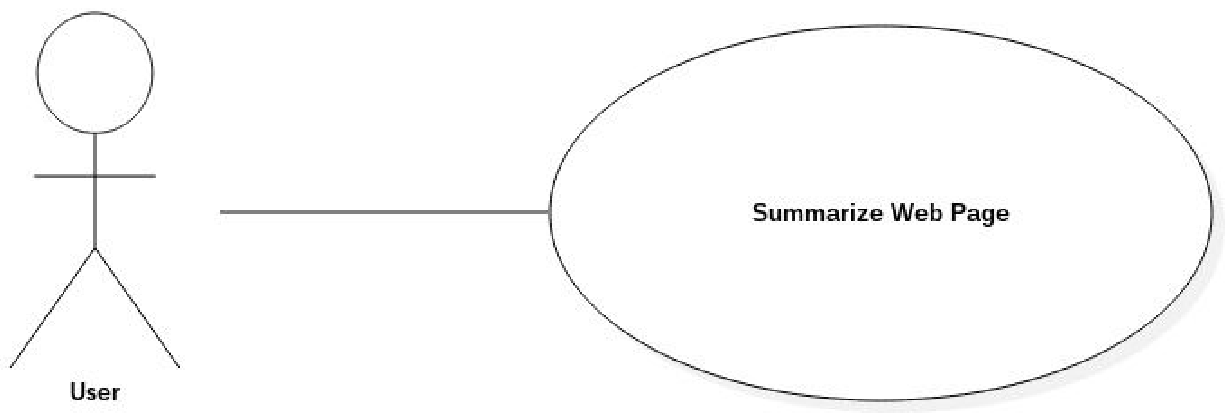

**Figure-1: Summarize Web Page Use Case Diagram**

| | |
|---|---|
| Use Case ID | 1 |
| Use Case | Summarize Web Page |
| Description | Getting summary of a webpage |
| Actor | User |
| Trigger | User clicks “Web Sayfasını Özetle” button in the extension pop-up. |
| Primary Scenario | - User clicks “Web Sayfasını Özetle” button.<br>- System takes the web page and sends it to the cloud server.<br>- Web page is parsed, and the summarization algorithm takes its body part as input.<br>- The algorithm gives summary sentences as output.<br>- Summary sentences are sent to the user’s browser page. |
| Exceptional Scenario | None |

- **Summarize File**

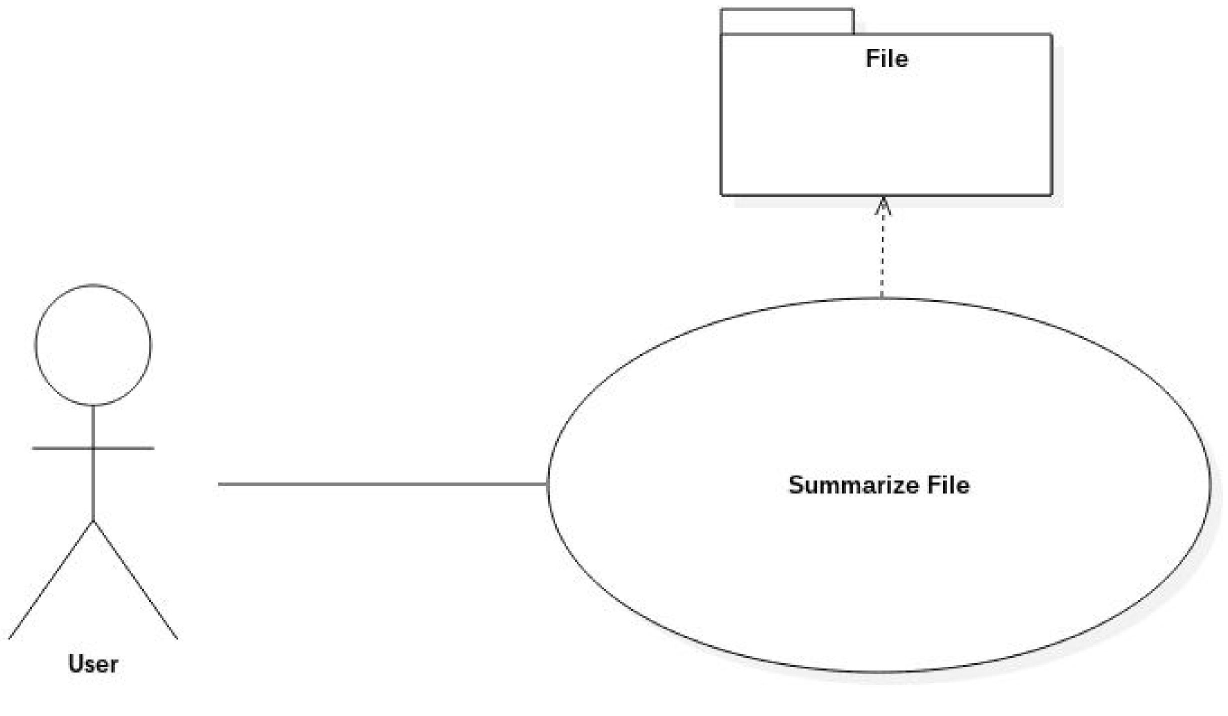

**Figure-2: Summarize File Use Case Diagram**

| | |
|---|---|
| Use Case ID | 2 |
| Use Case | Summarize File |
| Description | Getting summary with uploading a file |
| Actor | User |
| Trigger | User clicks “Dosyayı Özetle” button in the extension pop-up. |
| Primary Scenario | - User clicks “Dosyayı Özetle” button.<br>- System checks if the file is compatible.<br>- System takes the file and sends it to the cloud server.<br>[Exception: File is not compatible]<br>- The summarization algorithm takes the file as input.<br>- The algorithm gives summary sentences as output.<br>- Summary sentences are sent to the user. |
| Exceptional Scenario | - Error message is displayed.<br>- System wants another file from user. |

- **Summary Setting**

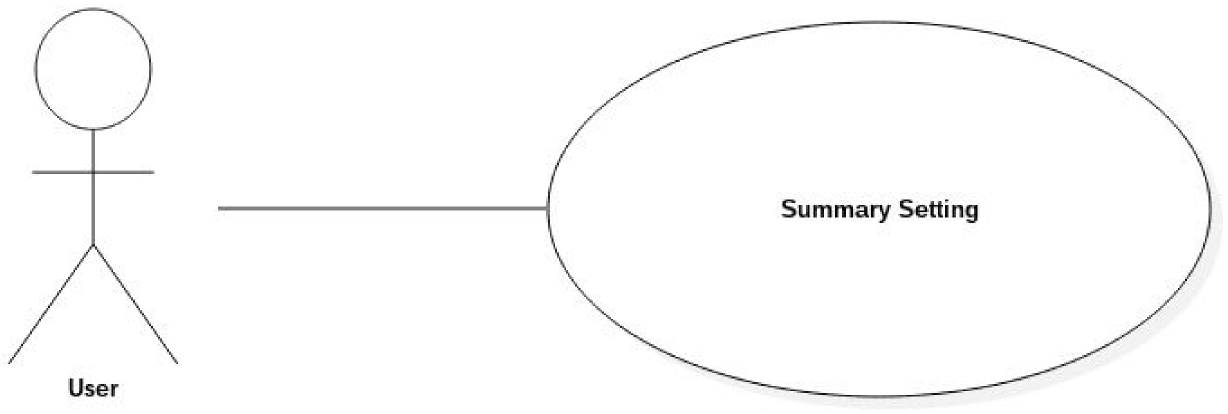

**Figure-3: Summary Setting Use Case Diagram**

| | |
|---|---|
| Use Case ID | 3 |
| Use Case | Summary Setting |
| Description | Setting the length of the summary |
| Actor | User |
| Trigger | User clicks “Ayarlar(Gear Logo)” button in the extension pop-up. |
| Primary Scenario | - User clicks “Ayarlar(Gear Logo)” button in the extension pop-up.<br>- User selects one through options. |
| Exceptional Scenario | None |

- **Train System**

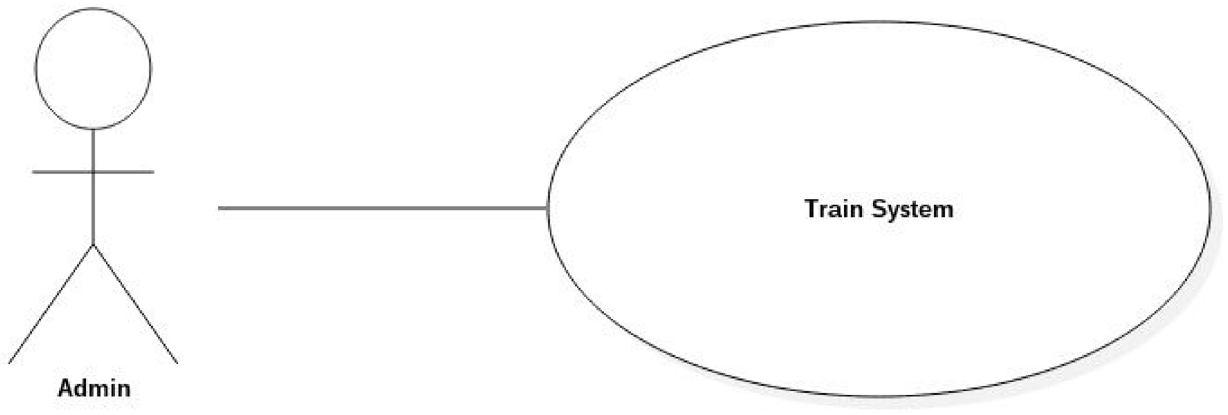

**Figure-4: Train System Use Case Diagram**

| | |
|---|---|
| Use Case ID | 4 |
| Use Case | Train System |
| Description | Training the system’s machine learning part for better result |
| Actor | Admin |
| Trigger | Admin enters the system. |
| Primary Scenario | - Admin enters the system.<br>- Trains Autoencoder and classifier with new datas. |
| Exceptional Scenario | None |

### 2.1.2 Actor Survey

- **User:** The user sends a request for the text to be summarized.
- **Admin:** Admin manages the website and configure a system to send responds to user requests. His/Her another role is to maintain the algorithm and the server.

## 2.2 Interfaces

### 2.2.1 User Interfaces

The user interfaces will be an icon in the browser. While using the browser, user's click will trigger a panel containing 3 buttons. With using these buttons user will decide if the text to be summarized from the current website or a text file from his/her hard drive.

The settings button will be used for determining the length of the sentence.

The prototype interface is the following:

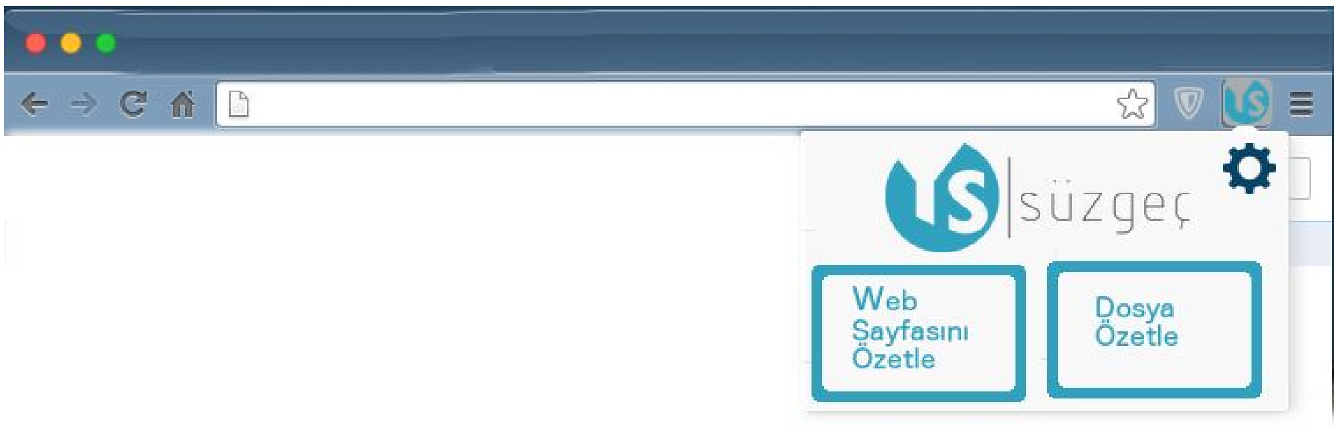

**Figure-5: Prototype Interface of Browser Extension**

After the selection of the text to be summarized it will be sent to the server and resulting package will be the summary of the text. The interface of the printed summary will be similar to the following image in the browser.

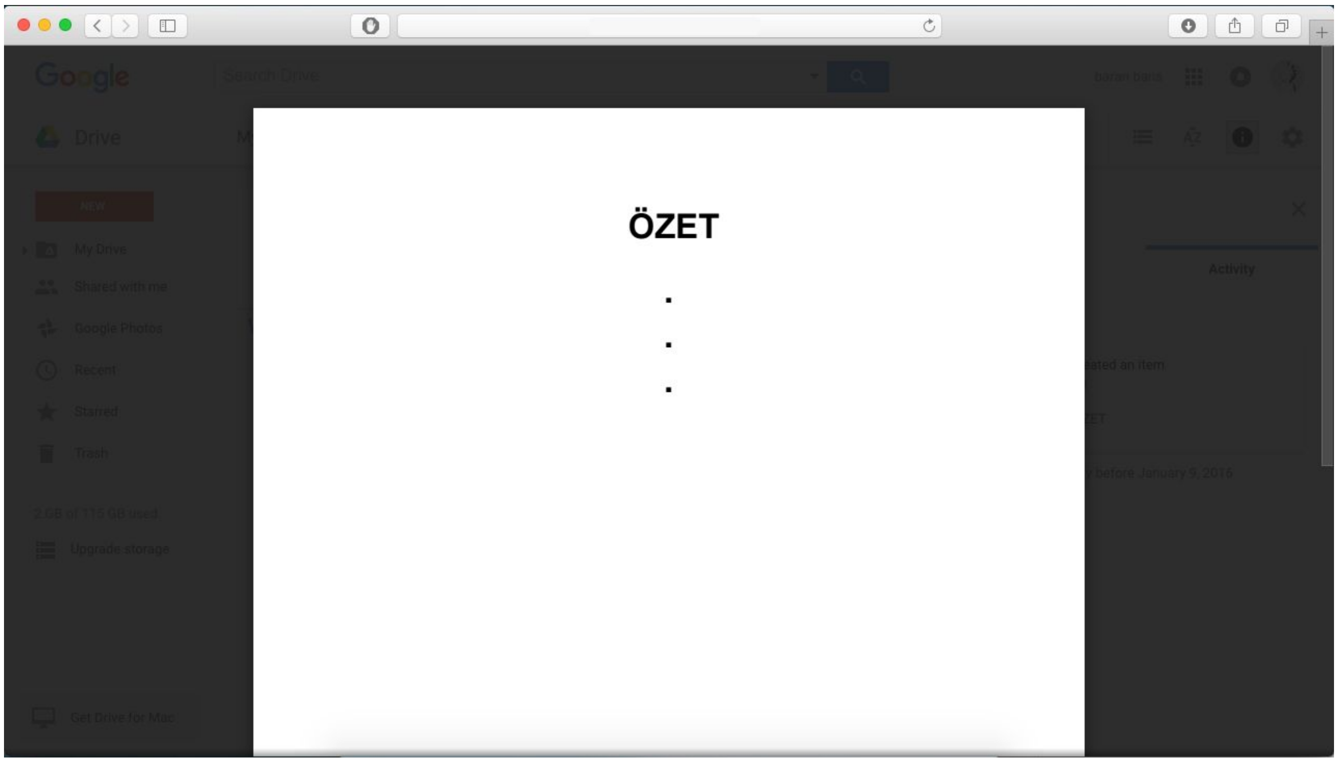

**Figure-6: Prototype Interface of Summary**

### 2.2.2 Hardware Interfaces

Not applicable.

### 2.2.3 Software Interfaces

In this system, there will be two APIs: TRNLTK and Zemberek NLP API.

TRNLTK will be used for tokenization and parsing texts to paragraphs, sentences and words. It is an NLP API.

Zemberek NLP API will also be used as mentioned. It will be used for getting words' roots, positions in sentence, stems and suffixes.

Both APIs are open source and specialized for Turkish language.

### 2.2.4 Communications Interfaces

The only communication is between the extension and the server. JQuery-AJAX will be used to send queries and receive ones. HTTP will be used as the protocol.

### 2.3 Constraints

- The browser extension must be implemented with Javascript.
- The components of the system must communicate well with each others.
- All 4 members will work for the project with no option to outsource.
- The server must run without a time constraint.
- The team members will be eager to learn deep learning concept.

## 3. Specific Requirements

### 3.1 Functional Requirements

As mentioned at section 2.1.1, these requirements are categorized by use cases. For any specific use case, there are specific requirements which are detailed below.

#### 3.1.1 Text Summarizer Requirements

- The system should provide text parser functions which can take the whole text and separate into sentences, paragraphs and words.
- The system should provide text-to-feature function which can take the necessary part and obtain a feature vector.
- The system should provide a well-trained Autoencoder to generate better inputs for classifier.
- The system needs a classifier which is well-trained to select summary sentences.
- The system should provide a sentence modifier to beautify and polish output text while changing some words with their synonyms etc.

#### 3.1.2 Summarize Web Page Requirements

- The system should provide a “Web Sayfasını Özetle” button with complete functionality. When clicked on this button, browser extension send the html of the current web page to the server.
- A function which detect body part and select text. This function needs to extract unnecessary text from html.
- The system should provide communication between server and client with necessary network functions such send and receive.

#### 3.1.3 Summarize File Requirements

- The system should provide a “Dosyayı Özetle” button with complete functionality. After user selected target file, the user presses the “Dosyayı Özetle” button and web page application send the file to the server.
- A set of functions provide the reading from file depends on file extension.
- The system should provide communication between server and client with necessary network functions such send file and receive file.

#### 3.1.4 Summary Setting Requirements

- The system should take parameters such as summary length from user before summarizing.

### 3.1.5 Train System Requirements

- The system should provide login screen for admin.
- The system should provide taking new data from admin to train Autoencoders or classifiers to improve reliability.

## 3.2 Non-Functional Requirements

### 3.2.1 Usability

The system should be easy to use. The user should reach the summarized text with one button press if possible. Because one of the software's features is timesaving.

The system also should be user friendly for admins because anyone can be admin instead of programmers. Training the Autoencoders and classifiers are used too many times, so it is better to make it easy.

### 3.2.2 Reliability

This software will be developed with machine learning, feature engineering and deep learning techniques. So, in this step there is no certain reliable percentage that is measurable.

Also, user provided data will be used to compare with result and measure reliability. With recent machine learning techniques, user gained data should be enough for reliability if enough data is obtained.

The maintenance period should not be a matter because the reliable version is always run on the server which allow users to access summarization. When admins want to update, it take long as upload and update time of executable on server. The users can be reach and use program at any time, so maintenance should not be a big issue.

### 3.2.3 Performance

Calculation time and response time should be as little as possible, because one of the software's features is timesaving. Whole cycle of summarizing a page/file should not be more than 30 seconds in order to 3 pages long document.

The capacity of servers should be as high as possible. Calculation and response times are very low, and this comes with that there can be so many sessions at the same times. The software only used in Turkey, than do not need to consider global sessions.

1 minute degradation of response time should be acceptable. The certain session limit also acceptable at early stages of development. It can be confirmed to user with "servers are not ready at this time" message.

### 3.2.4 Supportability

The system should require C, Java, Python and Matlab knowledge to maintenance. If any problem acquire in server side and deep learning methods, it requires code knowledge and deep learning background to solve. Client side problems should be fixed with an update and it also require code knowledge and network knowledge.

## 4 Data Model and Description

This section includes components and data objects of the system. Also a component diagram, a class diagram, and descriptions of components and classes can be found in this section.

### 4.1 Components

Our system will consist of 4 main components: Text parser, feature vector creator, Autoencoder and classifier.

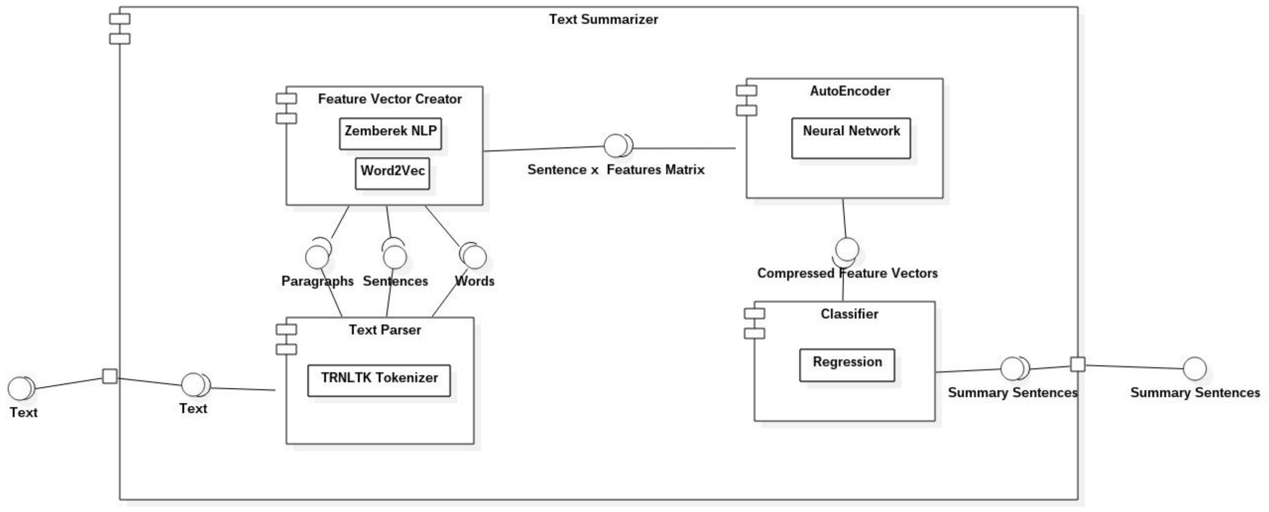

**Figure-7: Component Diagram of The System**

#### 4.1.1 Text Parser

In order to get sentence features, texts must be parsed into paragraphs and sentences. With the help of TRNLTK Tokenizer - this tool might be changed in the future provided that we find a better Turkish parser - we are going to parse the texts and send them to our feature vector creator.

### 4.1.2 Feature Vector Creator

Since we cannot simply give the sentences as an input to the Autoencoder, we have to create a proper sentence representation first. To do that, we are going to create a vector from all sentences’ features , e.g  and give it as an input to the Autoencoder. Feature vector creator is going to take each sentence as input and give their feature values as a vector output.

### 4.1.3 Autoencoder

Autoencoder is used in order to get a better representation and give it to the classifier. Autoencoder can be thought as the cells in our neural network and it builds the core of the deep learning. After training the Autoencoder, it will give compressed and better feature values for each sentence. With these representations, the classifier and the result -i.e summary- sentences will be more reliable.

This component is also trained with the inputs obtained from the sentences.

### 4.1.4 Classifier

Classifier is the last component of our system. It determines whether a sentence is a summary sentence or not.

Classifier uses the logistic regression method for classifying. The regression simply puts a line between the two classes which are the summary sentences and not summary sentences.

Classifier is going to be trained by us with the data we obtained from volunteers. After training with enough data sets, it will be ready to decide if a sentence is important or not.

## 4.2 Classes

There are 4 main classes in our project: Text, paragraph, sentence and word. Each of their fields and methods are going to be used in order to calculate the sentence features.

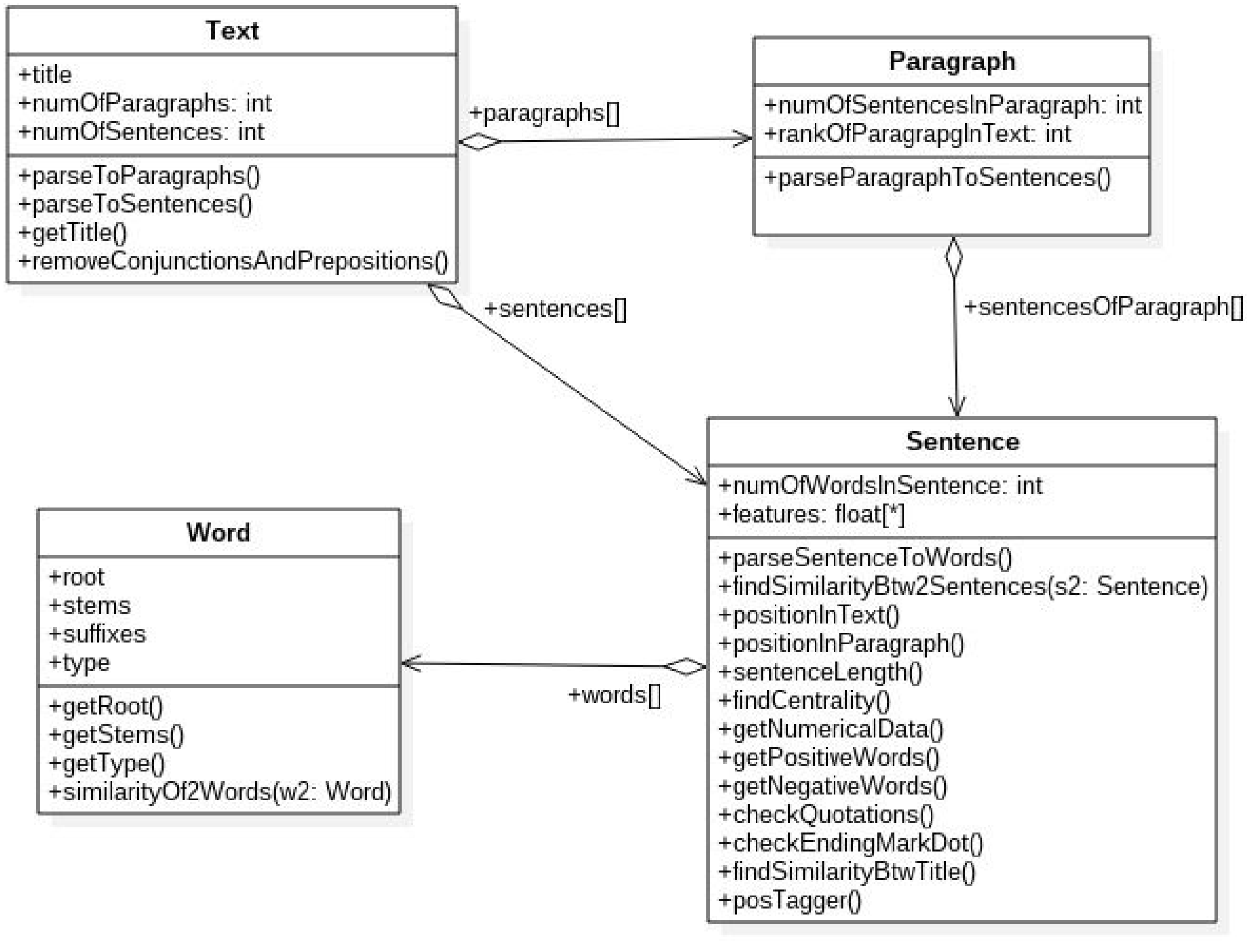

**Figure-8: Class Diagram of The System**

### 4.2.1 Text Class

Text class is the most complex class of the system. It has paragraphs, sentences and words. For dividing text to these parts, text class should have parser methods. Also there are number of sentences and number of paragraphs attribute in this class. These attributes are necessary for calculating sentence features.

### 4.2.2 Paragraph Class

Paragraph class is intermediary class of the system. In paragraph object, some necessary calculations are made for sentence features such as number of sentence in paragraph and rank of paragraph in text. It also has own parser to divide the paragraph in sentences.

#### 4.2.3 Sentence Class

Sentence class is the most important class of the system. Sentence object has methods to calculate feature values of itself with the information it takes from text, paragraph and word classes. It has a float list called “features”. “features” list has feature values of the sentence. The system combines “features” lists of the sentence objects of the text, and makes a features matrix with them. Autoencoder and Classifier components -mentioned in section 4.1- uses this features matrix. Sentence class also has own parser to divide the sentence in words.

#### 4.2.4 Word Class

Word class is the most basic class of the system. Using NLP APIs, we can get word’s root, stem and suffix parts, and type of the word such as verb or noun. Also using Word2Vec API, the cosine distance between two words can be calculated. These attributes are used for calculating sentence’s feature values.

### About this template

This template was adapted by Emre Akbas from two sources: the IEEE 830 [1] and the ``Modern SRS package” [2].

## 5 References

[1] IEEE Guide for Software Requirements Specifications," in IEEE Std 830-1984 , vol., no., pp.1-26, Feb. 10 1984, doi: 10.1109/IEEESTD.1984.119205,
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=278253&isnumber=6883

[2] Appendix C of Don Widrig, Dean Leffingwell, “Managing Software Requirements: A Unified Approach,”  Addison-Wesley Professional, Release Date: October 1999, ISBN: 0201615932.