BILKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CS 299

SUMMER TRAINING

REPORT

Ömer Oktay Gültekin

21901413

Performed at

FNSS SAVUNMA SİSTEMLERİ A.Ş.

20.06.2022 – 21.07.2022

Table of Contents

1 Introduction

Turkey has made significant progress in the defense industry in the past few decades. Therefore, I wanted to experience the work environment of one of the biggest defense companies in Turkey and decide whether it fits my expectations regarding the work environment, the work done, and my career development. As a second-year student, I only had a few choices to apply, so I interned at the IT department of one of the prestigious defense companies in Turkey, FNSS.

I expected to work on one of the apps or software used in the company's internal network and at least see how their vehicles' software is written and integrated. However, as a military zone regulation, most data were national security secrets and banned for interns and civilians. Therefore, the company was able to satisfy a small portion of my expectations.

Instead of working on the company's projects, our supervisor split us into groups of two and gave us a bunch of projects about the topic of our choice. That way, I had an opportunity to start working on natural language processing through projects designed to teach concepts and methods used in parts of the real projects developed in the company.

The rest of the report gives broader information about the company, my internship process, my work, the outcomes, and my conclusion about my internship.

2 Company Information

2.1 About the Company

FNSS, founded with the partnership of Nurol and FMC companies, is Turkey's first private defense industry company that became operational in 1989 by taking the order of 1698 armored combat vehicles [1]. The company has sold over 4000 products to many countries, including Turkey, Saudi Arabia, Oman, and Indonesia [1]. FNSS is not a mass production company; it is a project company that manufactures vehicles on demand. Therefore, the company, established to produce 1689 armored vehicles that do not require R&D, needed to establish an R&D department to make new products. When the 1689 armored vehicles were delivered, the R&D department was open ten years after the company was founded. That allowed the company to take on new projects and start producing most of its unique designs of tracked and wheeled armored combat vehicles, combat engineering vehicles, manned and unmanned turret systems, and unmanned ground vehicles [2]. The need for the information technologies department within the R&D department has increased, especially with the company's start to produce unmanned ground vehicles. For more information about the company, see the Appendices figure 2-7.

2.2 About the Department

The IT Department provides and manages the information technology infrastructure necessary for company-wide activities [3]. This department's responsibility is to provide and operate all kinds of equipment, such as computers, handheld terminals, printers, servers, backup units, (IT) security systems, wired/wireless network systems, etc. [3]. In FNSS, where design and production activities are of utmost importance, all software required to realize these operations,

especially e-mail service, office applications, ERP, PLM, and engineering software, are supplied, installed, and managed, and the needed end-user support is provided [3]. In addition, software development activities are carried out to meet the company's needs [3]. It is ensured that different departments can access the company's data whenever needed [3]. Business analytics systems required for analyzing and reporting this data are managed, and decision-making mechanisms are supported throughout the company [3]. The IT Department is responsible for securing any information produced and used within the FNSS processes [3]. The audit, authorization, and transactions required for data protection are followed up and analyzed [3]. The innovations needed for process improvements and increased productivity are also adapted to FNSS by the IT Department by closely following the developments in the field of technology [3].

2.3 About the Hardware and Software Systems

2.3.1 Hardware Systems

The company has two system rooms. One is for intra-company networks, computation power, and storage of the companies' electronic devices, whereas the other is for the company data available on the internet. With this separation, the company aimed to protect the sensitive data at the highest level and to minimize the damage in the event of a possible information leak. In addition, to further increase security, access to the system rooms of the internal and external network is given to only 2 and 5 personnel, respectively. The system rooms have high-tech Huawei modular data centers, Dell storage devices, routers, and switches. In addition to that, the internal system room has huge computation power. Apart from these, the company shuts down fifteen days a year to maintain the devices in the systems rooms, replace the old devices and make the necessary updates. The system administrator deletes the data on the replaced devices and makes sure that these devices will not be used by anyone again. As the last line of defense, professional hackers are called every month and asked to try hacking the devices in the system rooms.

Using virtualization technology, the system administrator allocates enough computing power and storage for each staff; in other words, all computers in the company are virtual machines. This method ensures that the hardware is used efficiently, that the system administrator can see what everyone is doing on their computer, and that he can instantly help anyone with computer-based problems. The company thus tries to minimize human-induced information leakage.

In addition, there are printers, fax machines, some programming and development cards in the office, CNC machines, and some advanced devices to test the vehicles in the production building and some embedded system hardware in the vehicles.

2.3.2 Software Systems

The software used by the company can be examined in three groups: the software they code, the software that is licensed, and the software that is licensed and modified.

Most of the company's own software are some utility programs aimed to be used by the company staff with mobile and desktop devices that aim to increase

work efficiency. The most significant example in this category is the Fnss App which enables the company personnel to see the meal list of the week, see the recent news about the company, and report the dangerous events witnessed by the staff. The second more helpful example in this category is the Service Table App which enables company staff to ask questions or ask for support from the person in the desired department. For these apps, the company primarily uses programming languages such as Java, PHP, C, C++, and Swift; frameworks such as .NET and Laravel. The company has also made it a principle to use new technologies with maximum benefits, such as using Azure in all services it provides. In addition, the company regularly adapts its current software to new technologies; for example, they are currently working on integrating Docker and Kubernetes technologies into all the existing software.

The company has also integrated much off-the-shelf software into its business environment. Most of the company's computers work in a Windows environment as many of the personnel are not programmers. The virtual machines are handled by VMware software. For coding, they use Jetbrains IDEs and Visual Studio; for embedded programming, they use Qt and the software with which the development cards are compatible. Other software is generally about production, such as the software they use to operate CNC machines. As a rule for using third-party software, the company pays attention not to use the latest version of external software unless necessary and to lag behind at least two versions.

Most of the programmers' time in the company is spent developing external software. Since the company's primary purpose is to produce armored vehicles, the company uses ERP systems for all kinds of external supply processes of these vehicles and uses engineering software such as CATIA and ANSYS for R&D works like designing the pieces of the armored vehicles. With the help of the Service Table App mentioned above, programmers are responsible for technical support and adding new features to these applications according to the staff's wishes.

In addition, the company does not provide internet access and computers to any interns to avoid legal action arising from unlicensed software that may be found on the interns' computers.

2.4 About my supervisor

- **Name:** Ferit Altay
- **Job Title:** Lead Software Engineer
- **Education:** BS in Computer Science, METU, 2014.
- **Email:** ferit.altay@fnss.com.tr

3 Work Done

3.1 Orientation

In a multi-department company, FNSS, all departments involve the different stages of the product life cycle; therefore, one must have enough knowledge about all departments to understand better what will be given and expected from one's department. That is why an employee who starts a job in FNSS receives extensive training. Therefore, the internship began with listening to a team lead from each department introducing their department and giving information about their place in

the product life cycle. In addition, we received seminars on workplace safety and the environment since FNSS is a workplace with high hazards and environmental impact. After we were told that we should not contact any chemicals and that we must have protective equipment when we entered the production building, the orientation part of the internship came to an end.

3.2 Examining the Software in Production

By the end of the fourth day, interns were split up into their departments. As the IT Department is a sub-department of R&D, we are given another seminar about the R&D part of the armored vehicles' lifecycle. Firstly, how they are designed and which software and techniques they are used were told. Secondly, all armored vehicles and their specific differences were introduced. Lastly, they explained the engineering challenges they faced while making product innovations.

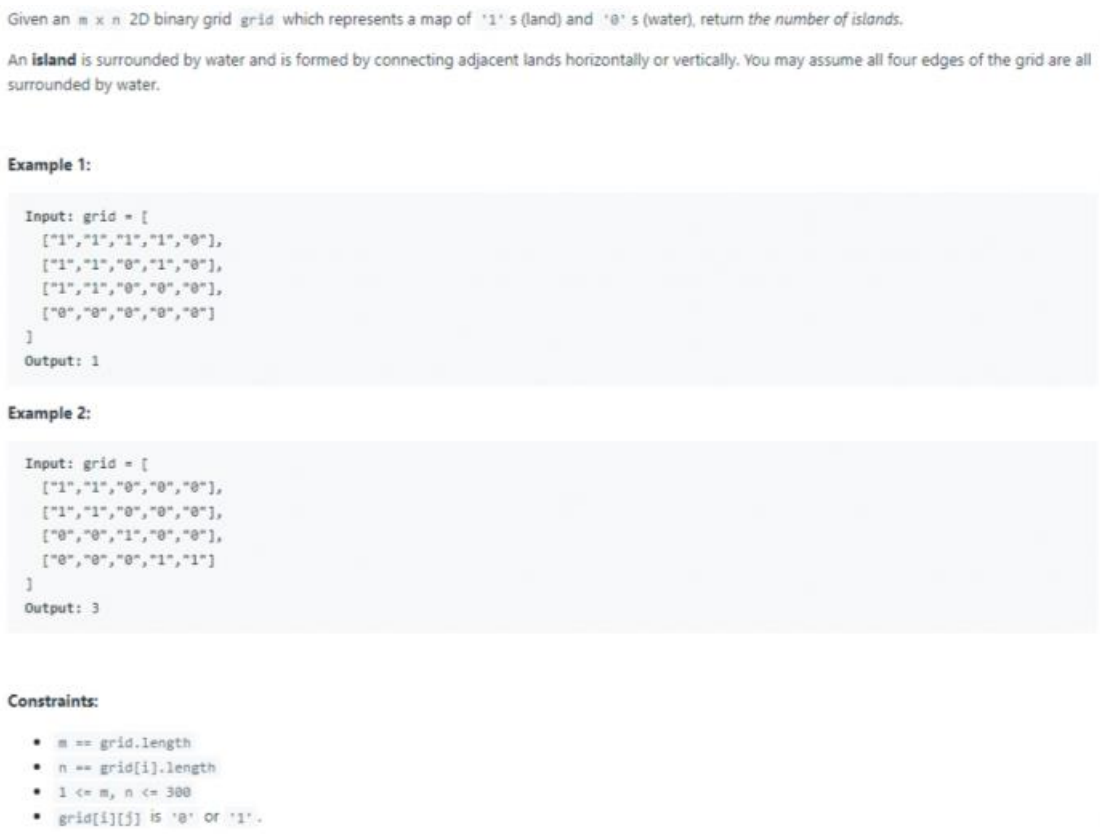
In the following days, I had a chance to wander in the production building to see how the production was done and how electronic devices were used during production. In this process, I especially tried to learn how the CNC machines work and operate with the code the production engineers wrote. In addition, I had the opportunity to see the interior and exterior of all vehicles and examine their digital components by making use of the preparation of all vehicles for a demo for company executives. Firstly, I saw the control panel of the vehicles. Secondly, I learned how they configure and embed the weapon system and other vehicle configurations by watching one of the Bilkent EEE graduate engineers. Finally, I watched the demo of the turret following a target traveling at 30 km/h and asked the engineers about the image processing and tracking algorithm of this tracking software. No further information I could give as this is all confidential information.

3.3 The First Days in the Office

The IT department in FNSS consists of many small teams. We were five interns and were all assigned to the company's PLM team, which primarily develops the company's ERP and engineering software. As their job was confidential, we could not do anything regarding company work. Therefore, they grouped us into three groups; one of them was one person. They gave us some areas and asked what field we wanted to do our projects in. I grouped with my friend from Bilkent, and we chose to work on Natural Language Processing. They wanted some time to think about the projects we could do and asked the company's System Administrator to care for us. He showed us the system rooms of the company and explained the devices inside. Then we spent one whole day with him to learn what system administrators do, how he uses VMware, how he gives technical support to the staff using virtual machines, and the advantages and disadvantages of being a system administrator. After he explained everything, he met us with a team whose job was to do the company's mobile and desktop apps. They also explained what they were doing, showed some codes from their projects, and, most importantly, explained why they use the Azure cloud service. They primarily emphasize that as a defense industry company, they faced many hacking attempts on their apps before using Azure. A vast part of the time was spent strengthening their apps' security. After using Azure, they could shift their attention from security to other things. Therefore, their work is optimized by using the latest technologies. I wrote most of section 2.3 from what I learned on the first days in the office.

3.4 Coding Done

Firstly, the supervisor states that they will prepare a report about us for the company that will be used in the potential job interview. To test our algorithmic skills, he gave us a problem in figure 1.



Given an `m x n` 2D binary grid `grid` which represents a map of `'1'`s (land) and `'0'`s (water), return *the number of islands*.

An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:

```
Input: grid = [
  ["1","1","1","1","0"],
  ["1","1","0","1","0"],
  ["1","1","0","0","0"],
  ["0","0","0","0","0"]
]
Output: 1
```

Example 2:

```
Input: grid = [
  ["1","1","0","0","0"],
  ["1","1","0","0","0"],
  ["0","0","1","0","0"],
  ["0","0","0","1","1"]
]
Output: 3
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 300`
- `grid[i][j]` is `'0'` or `'1'`.

Figure 1 Number of Islands [4]

After reviewing our answers, he gave us our first projects. We were told that we should use python in all projects that will be given, but we must figure out which modules to use ourselves.

3.4.1 The First Project: Simple Web Scraper

The purpose of this project was to enable us to learn the process of extracting data from the Internet, which is often used in natural language processing. See the Appendices figure 8-12 for the project code.

In the first stage of this project, we were given a link to an API (http://api.quotable.io/quotes/) that contains some quotes. The input would be provided by our supervisor and may only sometimes be correct. In this stage, we were expected to parse the URL, and if its status code is 200, we will output the content of the given quotes; else output the appropriate message. In this stage, we used the Python request library to get the content of the quotes in JSON form.

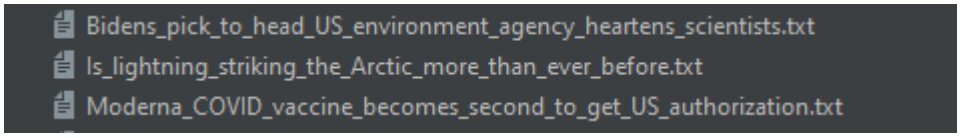
In the second stage, we were expected to parse an IMDB movie page and get the title and description of the chosen movie. As we inspected the movie pages,

we found that the title is stored in the h1 element, and the description is stored in the span element with the "data-testid": "plot-l" attribute. Therefore, since the simple request library will not be enough, we used the BeautifulSoup library to get the specific content we needed. The sample output for Star Wars: Empire Strikes Back movie:

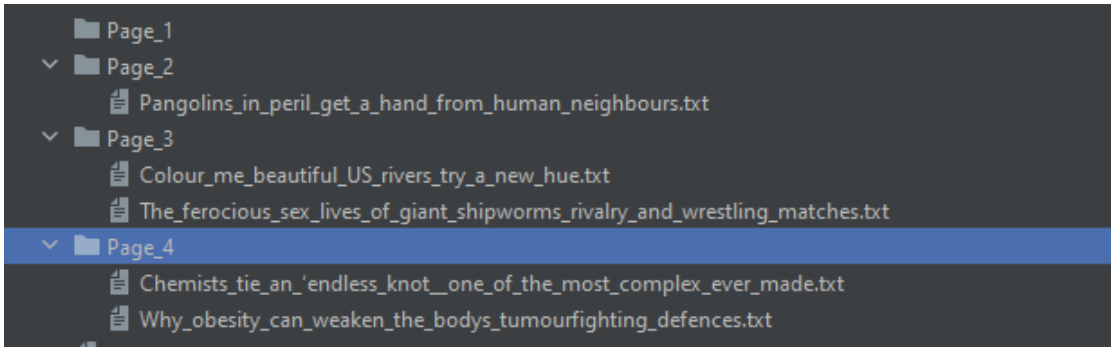
```
Input the URL:
https://www.imdb.com/title/tt0080684/
{'description': 'After the Rebels are brutally overpowered by the Empire on '
                'the ice planet Hoth, Luke Skywalker begins Jedi training with '
                'Yoda, while his friends are pursued across the galaxy by '
                'Darth Vader and bounty hunter Boba Fett.',
 'title': 'Star Wars: Episode V - The Empire Strikes Back'}

Process finished with exit code 0
```

In the third stage, we were told to save the downloaded site to a file if the site existed. This stage was for the preparation for future stages.

In the fourth stage, we were given an articles site URL that contained articles from 2020 and was expected to parse all the articles from the news category and save their contents into separate files. All unnecessary tags had to be removed, and a way to automatically enter the necessary article links was required. We dive into BeautifulSoup even further in this stage. Example output for the files created after running this stage code:



In the final stage, we were asked to parse the article topic from the given number of pages and create separate folders for every page. Example output for the folders created for the Research Highlight topic from the first four pages:



3.4.2 The Second Project: Simple Key Terms Extraction Tool

Keyword extraction is a standard Natural Language Processing (NLP) task used by search engines to show ads based on relevant keywords on a webpage. This project aimed to teach us the scientific ways of determining the importance of a

word in a text by creating a tool that extracts the most important words from the given text. See the Appendices figure 13-16 for the project code.

In the first stage of this project, an XML file that contained ten articles was given. At first, it is asked to find the five most frequent words to use as keywords for the given articles. It is given to use the NLTK library of Python to tokenize given text word by word. However, how to parse the XML file is left for research. We use the tokenize module of the NLTK library to tokenize the given articles and the etree module of the lxml library to parse the XML file. The output of the program for the first five articles:

```
Brain Disconnects During Sleep:
the of , . that

New Portuguese skull may be an early relative of Neandertals:
of the , in a

Living by the coast could improve mental health:
the to , . health

Did you knowingly commit a crime? Brain scans could tell:
the , . of a

Computer learns to detect skin cancer more accurately than doctors:
, the of . in
```

In the second stage, it was asked to eliminate all the punctuations and English's most frequent words that appear in the last stage's output. The task of figuring out how to convert all words into their root form was also given. In the solution, appropriate methods, and constants of NLTK were used to satisfy the given criteria. Specifically, the lemmatization process, which is used to get the root form of the words, was done by using the WordNetLemmatizer method of the NLTK library, which uses a dictionary to search for the given words and find their root corresponding root in the library while paying attention the word’s meaning in the context. The output of the program for the first five articles now becomes:

```
Brain Disconnects During Sleep:
sleep cortex consciousness say tononi

New Portuguese skull may be an early relative of Neandertals:
skull neandertal fossil europe year

Living by the coast could improve mental health:
health mental coast research people

Did you knowingly commit a crime? Brain scans could tell:
brain people say study wa

Computer learns to detect skin cancer more accurately than doctors:
dermatologist skin melanoma year cnn
```

In the third stage, it was asked to exclude all non-Noun words from the key terms extraction process by finding the corresponding NLTK method and its usage. The NLTK library already tags the words in the lemmatization process, and we just found the correct method to access the word tags. The output of the program for the first five articles now becomes:

```
Brain Disconnects During Sleep:
sleep cortex consciousness tononi activity

New Portuguese skull may be an early relative of Neandertals:
skull fossil europe year trait

Living by the coast could improve mental health:
health mental coast research living

Did you knowingly commit a crime? Brain scans could tell:
brain study wa suitcase result

Computer learns to detect skin cancer more accurately than doctors:
dermatologist skin melanoma year cnn
```

In the final stage, the first step was to learn the commonly used Term Frequency-Inverse Document Frequency (TF-IDF) approach to find the importance of a word in a text. The second step was to find the corresponding method from the sklearn library to calculate the TF-IDF metric of all the words, which will be used to find five keywords of the text. TF-IDF is a multiplication of TF and IDF values of the word [5]. The TF of the word is calculated by the number of words divided by the number of total words in the document [5]. The IDF of the word is calculated by the logarithm of the total documents divided by the number of documents containing that word [5]. TF-IDF is approached to 1 if the word is rare and 0 otherwise [5]. In this project, we used sentences as if they were documents. We use the TfidfVectorizer method of the sklearn library in this stage. The final output of the program for the first five articles:

```
Brain Disconnects During Sleep:
sleep cortex consciousness tononi tm

New Portuguese skull may be an early relative of Neandertals:
skull fossil europe trait genus

Living by the coast could improve mental health:
health coast mental living household

Did you knowingly commit a crime? Brain scans could tell:
brain suitcase study security scenario

Computer learns to detect skin cancer more accurately than doctors:
dermatologist skin melanoma cnn lesion
```

3.4.3 The Third Project: Simple Text Summarization Tool

The purpose of this project was to extend the knowledge previously gained for finding the keywords of the text to find √n sentences that summarize the given text, where n is the number of sentences in the text. We use the same XML file provided in the second project. See the Appendices figure 17-21 for the project code.

In the first stage, the following was asked: Parsing the file using the BeautifulSoup library, tokenizing the sentences, and printing the first n sentences. Unlike the previous project, we used the sent_tokenize method for the tokenization process. The first few words of the output of the program for the first four articles:

```
HEADER: Brain Disconnects During Sleep
TEXT: Scientists may have gained an important insight into the age-ol
Lines of communication between various parts of cerebral cortex - whi
Early neuroscientists assumed that consciousness wanes during sleep b
But electroencephalography (EEG) and other modern methods have since

HEADER: New Portuguese skull may be an early relative of Neandertals
TEXT: Half a million years ago, several different members of our genu
But which ones has been the subject of intense debate.

HEADER: Living by the coast could improve mental health
TEXT: People who live close to the sea have better mental health thar
According to scientists, living near the sea could support better men
Researchers from the University of Exeter used survey data from 25,96
After taking other related factors into account, they found that livi

HEADER: Did you knowingly commit a crime? Brain scans could tell
TEXT: The number of years someone spends behind bars can hinge on whe
But how is a judge or jury to know for sure?
A new study suggests brain scans can distinguish between hardcore cri
The study is unusual because it looks directly at the brains of peopl
Earlier research, including work by her, has instead generally looked
```

In the second stage, finding the most important sentences using the SumBasic algorithm was asked. The sumbasic algorithm is used to generate automatic text summaries and uses a basic idea of finding the frequently used words in the document [6]. The algorithm was quite complex, and we wrote our own method to implement this algorithm. Since the algorithm steps are quite long, they are not included in the report but are accessible in reference [6]. The first few words of the output of the program for the first four articles:

```
HEADER: Brain Disconnects During Sleep
TEXT: Early neuroscientists assumed that consciousness wanes during s
Communication between regions of the cortex might be one sign of this
During non-rapid eye movement (non-REM) sleep, the same TMS stimulus
An important follow-up, he says, will be to repeat the experiments du

HEADER: New Portuguese skull may be an early relative of Neandertals
TEXT: Found in 2014 in the Gruta da Aroeira cave in central Portugal
This new combination of features on a well-dated skull may help resea

HEADER: Living by the coast could improve mental health
TEXT: People who live close to the sea have better mental health thar
Those from low income households less than a kilometre from the coast
Published in the Health and Place journal, the findings suggest acces
The research used data from the Health Survey for England and compare

HEADER: Did you knowingly commit a crime? Brain scans could tell
TEXT: Those differing brain patterns only showed up when people were
Montague isn't sure why the results hinged on the order in which the
We don't know what to do with this,'' Montague says.
But another researcher who has used brain imaging to study criminal b
And he cautions that it's a very big leap from there to visions of 'e
```

In the third stage, finding the most important sentences using the TF-IDF approach was asked. NumPy arrays were used to calculate the TF-IDF metrics of the sentences. The first few words of the output of the program for the first four articles:

```
HEADER: Brain Disconnects During Sleep
TEXT: Early neuroscientists assumed that consciousness wanes during s
That left neuroscientists with a puzzle: If the brain is still active
Tononi has spent years developing a theory that equates consciousness
'We would predict a pattern which is much more similar to wakefulness

HEADER: New Portuguese skull may be an early relative of Neandertals
TEXT: But which ones has been the subject of intense debate.
A newly discovered partial skull is offering another clue to help sol

HEADER: Living by the coast could improve mental health
TEXT: According to scientists, living near the sea could support bett
Researchers from the University of Exeter used survey data from 25,96
The research used data from the Health Survey for England and compare
Researchers say their findings add to the growing evidence that acces

HEADER: Did you knowingly commit a crime? Brain scans could tell
TEXT: But how is a judge or jury to know for sure?
In some cases, the people knew for certain they had contraband in a s
But there was an unexpected twist.
'I'm a scientist, so I was like, 'This is the most interesting part o
We don't know what to do with this,'' Montague says.
```

In the final stage, it is asked to reduce the TF-IDF scores of words containing the words in the title by multiplying it with a constant and to reduce the probability of sentences containing these words being found in summary. Numpy arrays were used to find the mean value of the words in sentences after updating the scores according to the words in the title. The first few words of the output of the program for the first four articles:

```
HEADER: Brain Disconnects During Sleep
TEXT: Scientists may have gained an important insight into the age-ol
Early neuroscientists assumed that consciousness wanes during sleep b
That left neuroscientists with a puzzle: If the brain is still active
'We would predict a pattern which is much more similar to wakefulness

HEADER: New Portuguese skull may be an early relative of Neandertals
TEXT: But which ones has been the subject of intense debate.
A newly discovered partial skull is offering another clue to help sol

HEADER: Living by the coast could improve mental health
TEXT: According to scientists, living near the sea could support bett
Published in the Health and Place journal, the findings suggest acces
The research used data from the Health Survey for England and compare
Researchers say their findings add to the growing evidence that acces

HEADER: Did you knowingly commit a crime? Brain scans could tell
TEXT: But how is a judge or jury to know for sure?
In some cases, the people knew for certain they had contraband in a s
But there was an unexpected twist.
'I'm a scientist, so I was like, 'This is the most interesting part o
We don't know what to do with this,'' Montague says.
```

3.4.4 The Fourth Project: Simple Text Generator

This project aimed to create a tool that analyzes any given corpus and creates a given number of sentences by predicting the next word from the previous one. The Game of Thrones Series script was given to train the program in this project. See the Appendices figure 22-26 for the project code.

In the first stage, the number of all and unique tokens in the script was asked. Since the following stages will try to create sentences, the corpus split based on whitespaces, and the token number was found by the FreqDist module of the NLTK library. The output of the program in this stage was:

```
corpus.txt
Corpus statistics
All tokens: 287968
Unique tokens: 21262
```

In the second stage, it was asked to find the number of bigrams (two adjacent tokens) in the corpus since the sequence of words usually is more meaningful than the randomly selected words. The NLTK library's corresponding method was used to

find the number of bigrams. The sample output that outputs the bigram pairs for the test program:

```
corpus.txt
Number of bigrams: 287967

0
Head: What     Tail: do
4
Head: They're     Tail: savages.
5
Head: savages.     Tail: One
3000
Head: can     Tail: protect
```

In the third stage, it was asked to create a Markov Chain Model, a statistical model in which the probability of every event depends only on the previous event. To check this stage, it is asked to print the three most probable tails for the given word. The sample output for the test program:

```
King
Head: King
Tail: in     Count: 76
Tail: of     Count: 24
Tail: Robert     Count: 15

North
Head: North
Tail: is     Count: 11
Tail: of     Count: 8
Tail: has     Count: 5

night
Head: night
Tail: is     Count: 20
Tail: to     Count: 8
Tail: and     Count: 8
```

In the fourth stage, it was asked to form ten sentences consisting of ten words, starting with a randomly chosen word among the words in the corpus. random.random.choices() method was used to predict the next word since it can consider the weights of the given words in the process. The sample output for the test program:

```
Mmmm, very much. For the best archer in your fire
fire engulfs the same sort of the lands and second
second thoughts. But the death I can stop him. I
I wish I can't touch of art, really. We just…
just… It's your return, you'll demand a man alive. Which
Which is the difference between life to trade your crimes?
crimes? Yes, we've allowed Robert Baratheon. They told me- How
How do you are, you'll face collapse into the reason
reason half a reason? Is this gift from the fields
fields are a whole life with his brothers. No, but
```

In the fifth stage, it was asked to form ten sentences consisting of at least five words, with the rule that all sentences start with a capital word and end with a punctuation symbol. The sample output for the test program:

```
Catelyn Stark. I had one, isn't it?
Bread, Your Grace. A good as I'm afraid.
Silent Sisters? With all its transportation himself.
Momma, momma. Maybe she whispers of course.
Smith to undermine me. Come on!
Council I'm going to the baby into King's Hand was as anyone else.
Striking your brother. No, you advise him.
Cripples and you die sword shall I had her in a child?
Power to be human. Oh, not quite fancied him.
Yarwyck, Lord Commander, one point where you fool!
```

In the final stage, it was asked to make the model based on trigrams, not bigrams, so that the coming word would be predicted based on the previous two words to make the sentences more realistic. The trigrams module of the NLTK library was used instead of bigrams. The sample output for the test program:

```
Good news or bad? Both.
The tide's against us. I'm a Tarley.
Dany, tell them, make them behave.
Watch have sworn those were shark fins.
Your Uncle Brandon. Your handsome, arrogant, cruel Uncle Brandon.
Train, then. Learn to fight for you?
Cersei, giving up that easily.
Beyond tonight, I beg you.
Walda to her tent. She's tired from the Kingslayer.
That it was all I know how you feel.
```

3.4.5 The Fifth Project: Simple Opinion Detector

This project aimed to create a tool that analyzes the SAR14 corpus that includes 234000 IMDB movie reviews with scores of 1 to 10. The tool deduces the number of conclusions from the number of positive and negative reviews. See the Appendices figure 27-31 for the project code.

In the first stage, we preprocess the first 150000 reviews from the given data to use in further stages. This stage is aimed at us getting accustomed to the usage of Pandas DataFrame. As an output, the program prints the head and tail of the data frame as shown below:

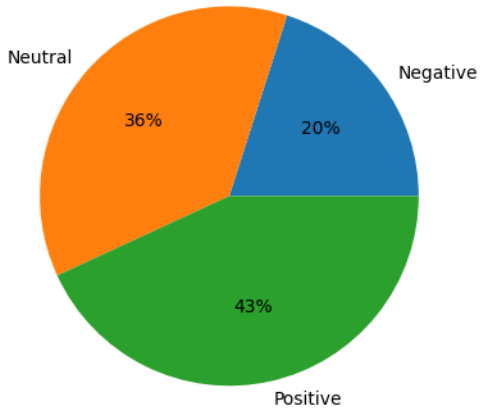
```
                                                Review Score
0  " The first art-film ? . this is possibly the ...    10
1  " collision between the tradition of family va...    10
2  " An everyday occurrence for posterity . Louis...     4
3  " The first family film ? . Monsieur Lumi re n...     4
4  " Ouch ! That 's Got ta Hurt ! -LRB- SPOILERS ...     8
                                                Review Score
149995  " The Best Movie of 2003 . Final Score : 10 -L...    10
149996  " big disappointment . I was excited to watch ...     2
149997  " Unforgettable . This is one of the most beau...     9
149998  " Tokyo stars in tedious film . The only reaso...     3
149999  " Zen style . - Slight spoilers are included -...     8
```

In the second stage, tokenization and lemmatization were done as in the previous projects. However, to get rid of the punctuation marks and digits, the regexp_tokenize method was used, which was not used in earlier projects. As an output, the program prints the head and tail of the lemmatized data frame as shown below:

```
0    [first, art, film, possibly, beautiful, early,...
1    [collision, tradition, family, value, modernit...
2    [everyday, occurrence, posterity, louis, lumie...
3    [first, family, film, monsieur, lumi, want, u,...
4    [ouch, got, ta, hurt, lrb, spoiler, rrb, spoil...
Name: Lemmas, dtype: object
149995    [best, movie, final, score, lrb, rrb, open, bo...
149996    [big, disappointment, excited, watch, movie, s...
149997    [unforgettable, one, beautiful, love, story, e...
149998    [tokyo, star, tedious, film, reason, spending,...
149999    [zen, style, slight, spoiler, included, langua...
Name: Lemmas, dtype: object
```

In the third stage, the corpus's sentiment analysis was done using a lexicon-based approach. We found positive and negative words collected by the NLP researchers and gave each sentence a sentiment score based on how many words the sentence has from the two lists. Specifically, positive and negative words add 1 and 0 to the sentiment score, respectively. Then each sentence is tagged as Neutral if the sentiment score is between -3 and 3, Positive if the score is greater than 3, and Negative if the score is less than -3. After that, we visualized the result using the matplotlib library and output the class with the most reviews, the review percent of the rarest class, and the percentage difference between Positive and Negative reviews. Pandas series and matplotlib methods were practiced in this stage. The output and graph of this stage are shown below:

```
The answer to the 1st question: Positive
The answer to the 2nd question: 20%
The answer to the 3rd question: 23%
```



The fourth stage aimed to prevent dependence on the dictionary of positive and negative words. Instead, we tried integrating machine learning into the tool in this and the following stages. In this stage, more effective sentiment analysis was done using the pandas series map method, and the analysis was vectorized using the TF-IDF approach. Finally, the vectorized data is split into training and test sets such that 75 percent of the randomly selected data is in the training set and remains in the test set. The output that answers some of the questions asked in this stage is shown below:

```
The size of the TF-IDF dictionary is 142850 words.
The size of the training set is 112500 rows.
The shape of the matrix of test data is (37500, 142850).
```

In the final stage, we used SGDClassifier from sklearn, which is generally used for sentiment analysis tasks. We fed the classifier with the previously vectorized data. In that stage, more accurate sentiment analysis results for the Negative and

Positive tags were found. The output of the classification_report method of the sklearn library is shown below:

	precision	recall	f1-score	support
Negative	0.77	0.65	0.70	5775
Positive	0.76	0.80	0.78	15585
accuracy			0.80	21360
macro avg	0.77	0.72	0.74	21360
weighted avg	0.75	0.70	0.72	21360

3.5 Final Presentation

In the last week of the internship, the company asked each intern to make a group presentation on a technical subject so that our presentation skills could be measured and included in our files. We presented design patterns in front of the HR department and all interns. I presented Singleton, Composite, and Template Design Patterns in that presentation. On the remaining days, we watch other interns' presentations and attend a certification ceremony on the last day.

4 Performance and Outcomes

4.1 Solving Complex Engineering Problems

Each project I have done in my training has its own complex engineering problems. In the first project, my teammate and I needed to parse the given page, find the articles with the desired titles, and parse their content from their page. It was pretty complex to figure out how they store needed data in thousands of lines of code. I could only use a little of the education I received from the university since I had only finished the second class. However, the algorithm skills I acquired at the university and finding needed information in official docs that I gained while doing homework in school eased the process for me. In other projects, I faced some complex engineering problems that the Natural Language Processing area has and used standard methods to solve them. For example, getting the root form of every word in a million-word corpus was a challenging problem, but I used the common solution of the WordNetLemmatizer method of the NLTK library. In other words, in my internship, I learned a lot of complex engineering problems in the domain of NLP and industry-standard solutions for them.

4.2 Recognizing Ethical and Professional Responsibilities

The defense company FNSS expects high professional responsibilities to obey military zone regulations from all its personnel. Therefore, I was expected not to take a photo or even mention anything confidential I saw in the company. Fnss highly enforces the professional rules they have, even on the interns. The employees could not rest in a place visible from the company's workplace during working hours. For example, I warned about working in a leisure place during working hours on a sunny day.

The company has flexible working hours, but one must fulfill the total working hours in a month. I saw an engineer that did not work a week before and tried to complete working hours that week, but she was talking on the phone with her friends when she was supposed to do the company work. Therefore, working hours are more ethical responsibility than obligation.

Regarding engineering responsibility, we were supposed to join a meeting with our supervisor at the end of every stage of the given projects. We were expected to have an ethical responsibility of contributing equally to the projects we did in the group of two. Also, we had a professional duty to inform our supervisor when we were stuck.

4.3 Making Informed Judgments

I made informed judgments mainly about the balance of efficiency and reliability. For example, stemming and lemmatization are two methods for getting the word's root form. Stemming has the advantage of efficiency because of deleting every character one by one from the end of the term until reaching something meaningful. In contrast, lemmatization gets the word's root form considering the context of the word with a disadvantage of efficiency but with the advantage of reliability that the root forms are mostly correct. Considering the domain and the use cases of the program, I made informed judgments to use the lemmatization technique since the projects that can take long processing times will be used primarily for one specific file, and the output will be used for a long time. Therefore, accuracy is more important in the usage context. As a result, the last project takes approximately three minutes to process the 150000 movie reviews but gives correct results above 90% thanks to lemmatization and the machine learning module, sklearn.

4.4 Acquiring New Knowledge by Using Appropriate Learning Strategies

Since Natural Language Processing (NLP) was new for both my teammate and me, we used various learning strategies to learn the standard solutions in the NLP domain. While I searched the problems we encountered in official docs of the NLTK library of python and programming forums like Stackoverflow, my teammate mostly spent time on the tutorials on Youtube. That way, we gained both practical and theoretical information about the subject.

4.5 Applying New Knowledge as Needed

Our projects were quite complex; thus, we needed to work outside the company with my teammate. Therefore, we used the Code With Me tool of the Pycharm IDE to write the codes together. Apart from that, we learned the various concepts used in NLP, such as the Sumbasic algorithm, Markov Chain Model, and TF-IDF metric approach, using appropriate learning strategies mentioned in section 4.4. Applying these concepts, we developed our projects that use intuitive approaches to solve the given problems to use more industry-standard solutions. We selected the most appropriate candidate for those solutions by making informed judgments.

4.6 Awareness About Diversity, Equity, and Inclusion

FNSS has a wide diversity, mainly due to the company's size. Since they produce the vehicles from the beginning, there are many departments in FNSS. All departments require different types of personnel. The company's staff are mainly split into two types: blue and white-collar workers.
The blue collars are generally socioeconomically lower than white collars. However, it is essential to note that a person with no prior experience and little education can start a job as a blue-collar worker in FNSS. FNSS has diversity in age groups, from newly graduated engineers to retired military personnel testing the vehicle's capabilities. They are an international company and have offices in four other countries. Therefore, the language in the work environment varies as sometimes an employee comes from another country. When an R&D department develops something new, they generally send some staff abroad or call experts from other countries, contributing to the company's diversity. Even though there is no discrimination in the job application criteria regarding gender, the number of male staff is far more than females.

In terms of equity, all staff are welcome to use the facilities equally.
The company provides lunch and dinner to all staff for free, and there is free transportation for all. Even the company's CEO gets in line and gets his food for meals.

In terms of inclusivity, FNSS needs to develop itself. A CNC machine operator who works all day in front of the machine cannot interfere with the code the production engineer sends, even if he knows better ways of doing the job.

For more information about the company's diversity, see the Appendices figure 2-7.

5 Conclusions

Being an intern at FNSS was a good experience in getting to know Turkish defense industry companies. Since each project file is available for in-house personnel to read, I had a chance to see all stages of the production of a vehicle from scratch to the final, both on file and in reality. In the files, I saw how they solved the engineering problems unknown to them in the R&D part. First, by calling help from university professors; second, they buy books; and third, they call experts from abroad. All enlightened me on how to divide a big project into small parts and conquer smaller problems. Seeing how a multi-disciplinary project was managed is important to me since I want to establish an innovative company that makes products consisting of software and hardware together.

From a computer science student perspective, I first learned how instructional types of code are written for production machines. Secondly, I examined the military vehicles' digital components and learned how they embed code into the vehicles. Thirdly, I understand the importance of the latest technologies, such as Azure and Kubernetes. Fourthly, even though I did not have any prior experience with the network, seeing a billion dollars worth of system room in reality with some explanations teaches me a lot. Finally, I learned what web scraping is and how it is done; and I started learning natural language processing techniques with Python NLTK libraries.

6 References

[1] “Tarihçemiz”.
https://www.fnss.com.tr/tr/kurumsal/fnss-tarihcesi. [Accessed: Oct 1, 2022].

[2] “Products.”
https://www.fnss.com.tr/en/products. [Accessed: Oct 1, 2022].

[3] “FNSS STAJYER BİLGİLENDİRME DOSYASI / 2022”.
https://drive.google.com/file/d/1a0NsizXxQw_py0c1WSBZz6fKXsLgefcx/view?usp=sharing. [Accessed: Oct 1, 2022].

[4] “Number of Islands.”
https://leetcode.com/problems/number-of-islands/. [Accessed: Oct 3, 2022].

[5] “Understanding TF-ID: A Simple Introduction.”
https://monkeylearn.com/blog/what-is-tf-idf/. [Accessed: Oct 23, 2022].

[6] “SumBasic algorithm for text summarization.”
https://iq.opengenus.org/sumbasic-algorithm-for-text-summarization/. [Accessed: Oct 23, 2022].

7 Appendices

The Design Patterns presentation can be accessed from this link: https://docs.google.com/presentation/d/1O2z-qUthnzANHC9HUmphpN0Cv_klxVXe-rFV5j5FyEs/edit?usp=sharing.

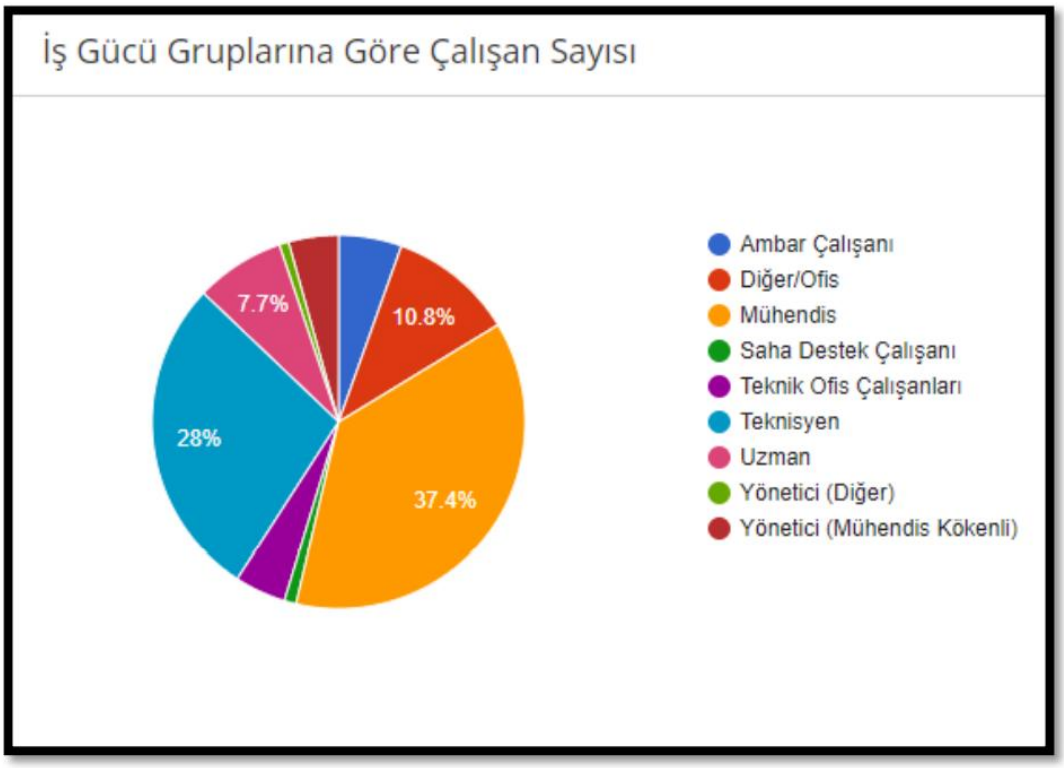


Figure 2 Number of Employees by Workforce Groups [3]

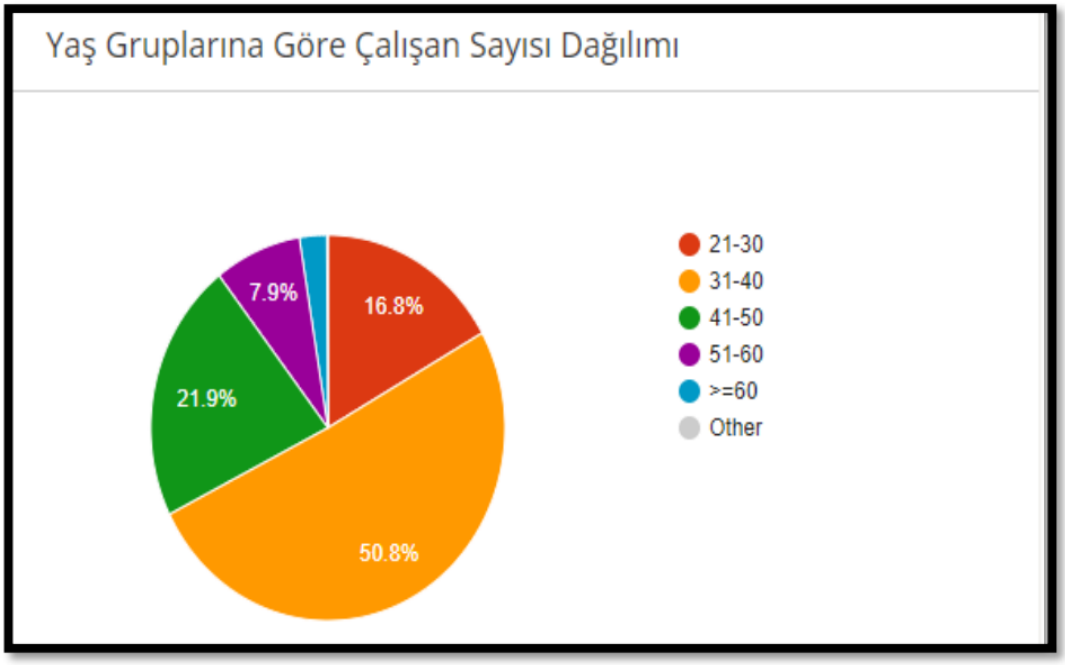


Figure 3 Number of Employees by Age Groups [3]

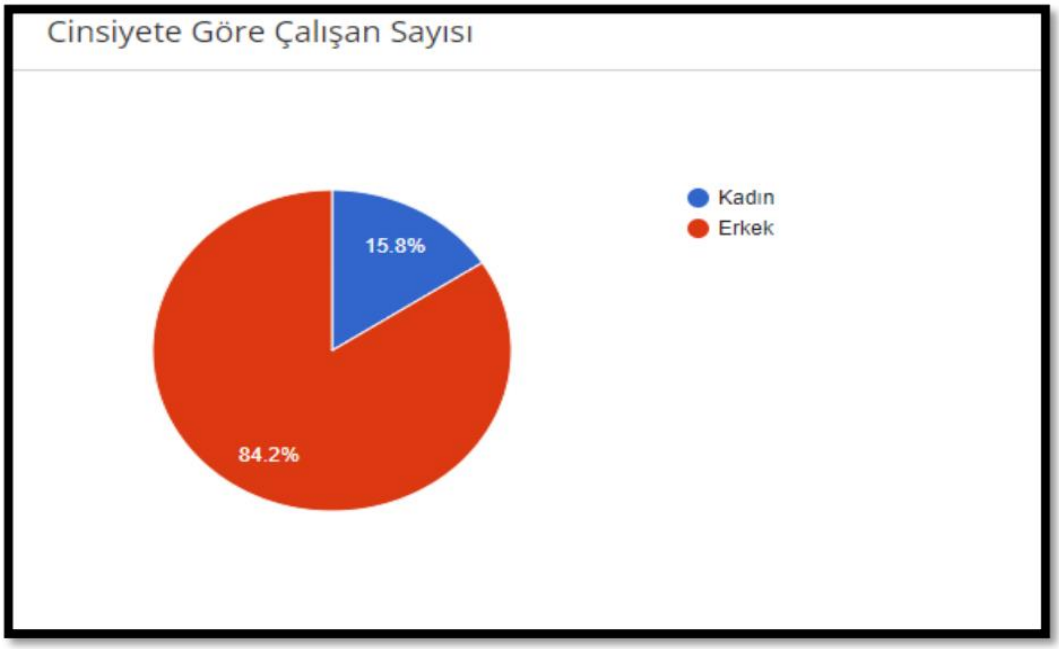


Figure 4 Number of Employees by Gender Groups [3]

Education Level	Sayı
DOKTORA	6
YÜKSEK LİSANS	178
LİSANS	370
YÜKSEKOKUL / ÖN LİSANS	87
MESLEK LİSESİ	226
LİSE	57
İLKÖĞRETİM	45

Figure 5 Number of Employees by Education Levels [3]

Departman Adı	Çalışan Sayısı
Kaynak / Welding	49
Talaşlı / Machining	14
Montaj / Assembly	63
Boya / Painting	28
Tedarik Zinciri Yönetimi / Supply Chain Management	57
Planlama ve Envanter Yönetimi / Planning and Inventory Maa	83
Üretim ve Planlama / Production & Scheduling	16
Üretim Mühendisliği / Manufacturing Engineering	25
Kalite Yönetimi ve İş Mükemmelliği / Quality Management & Business Excellence	83
Orta ve Uzak Doğu Programlar / Middle and Far East Programs	11
Mühendislik ve AR-GE / Engineering and R&D	291
Tesis ve Bakım / Facilities and Maintanence	33
Ürün Destek / Product Support	42
Pazarlama ve Programlar Grup Başkanlığı / Marketing & Programs	14
Üretim ve Tesisler / Production and Facilitie	18
Genel Müdürlük / General Management	6
Finans / Finance	33
İnsan Kaynakları / Human Resources	70
Bilgi Teknolojileri / Information Technologies	33

Figure 6 Number of Employees by Department [3]

Bölüm	Mühendislik Alanı	Çalışan Sayısı
BİLGİ TEKNOLOJİLERİ	BİLGİSAYAR	4
BİLGİ TEKNOLOJİLERİ	ELEKTRİK – ELEKTRONİK	2
BİLGİ TEKNOLOJİLERİ	ENDÜSTRİ	6
BİLGİ TEKNOLOJİLERİ	HAVACILIK VE UZAY	1
BİLGİ TEKNOLOJİLERİ	MAKİNA	8
FİNANS	MEKATRONİK	1
GENEL MÜDÜRLÜK	ELEKTRONİK VE İLETİŞİM SİSTEMLERİ	2
GENEL MÜDÜRLÜK	JEOLOJİ	1
GENEL MÜDÜRLÜK	MAKİNA	1
GENEL MÜDÜRLÜK	METALURJİ	1
İNSAN KAYNAKLAR	JEOLOJİ	1
İNSAN KAYNAKLARI	ENDÜSTRİ	3
İNSAN KAYNAKLARI	MADEN	1
İNSAN KAYNAKLARI	MAKİNA	1
KALİTE YÖNETİMİ VE İŞ MÜKEMMELLİĞİ	ELEKTRİK ELEKTRONİK	2
KALİTE YÖNETİMİ VE İŞ MÜKEMMELLİĞİ	ENDÜSTRİ	3
KALİTE YÖNETİMİ VE İŞ MÜKEMMELLİĞİ	GENETİK VE BİYOMÜHENDİSLİK	1
KALİTE YÖNETİMİ VE İŞ MÜKEMMELLİĞİ	KİMYA	1
KALİTE YÖNETİMİ VE İŞ MÜKEMMELLİĞİ	MAKİNA	16
KALİTE YÖNETİMİ VE İŞ MÜKEMMELLİĞİ	METALURJİ VE MALZEME	6
KALİTE YÖNETİMİ VE İŞ MÜKEMMELLİĞİ	METALURJİ	4
KALİTE YÖNETİMİ VE İŞ MÜKEMMELLİĞİ	OTOMOTİV	1
TEKNOLOJİ BAŞKANLIĞI	BİLGİSAYAR	1
TEKNOLOJİ BAŞKANLIĞI	ELEKTRİK ELEKTRONİK	21
TEKNOLOJİ BAŞKANLIĞI	ELEKTRİK	1
TEKNOLOJİ BAŞKANLIĞI	ELEKTRONİK	3
TEKNOLOJİ BAŞKANLIĞI	ENDÜSTRİ	14
TEKNOLOJİ BAŞKANLIĞI	HAVACILIK	3
TEKNOLOJİ BAŞKANLIĞI	HAVACILIK VE UZAY	4
TEKNOLOJİ BAŞKANLIĞI	İMALAT	3
TEKNOLOJİ BAŞKANLIĞI	İNŞAAT	2
TEKNOLOJİ BAŞKANLIĞI	KİMYA	2
TEKNOLOJİ BAŞKANLIĞI	KONTROL VE OTOMASYON	1
TEKNOLOJİ BAŞKANLIĞI	MAKİNA	131
TEKNOLOJİ BAŞKANLIĞI	MEKATRONİK	3
TEKNOLOJİ BAŞKANLIĞI	METALURJİ VE MALZEME	5
TEKNOLOJİ BAŞKANLIĞI	OTOMOTİV	8
TEKNOLOJİ BAŞKANLIĞI	SİSTEM	2

ORTA VE UZAK DOĞU PROGRAMLAR	ENDÜSTRİ	1
ORTA VE UZAK DOĞU PROGRAMLAR	ENDÜSTRİ	5
ORTA VE UZAK DOĞU PROGRAMLAR	MADEN	1
ORTA VE UZAK DOĞU PROGRAMLAR	MAKİNA	9
ORTA VE UZAK DOĞU PROGRAMLAR	METALURJİ	2
PAZARLAMA VE PROGRAMLAR	ELEKTRİK ELEKTRONİK	1
PAZARLAMA VE PROGRAMLAR	ENDÜSTRİ	9
PAZARLAMA VE PROGRAMLAR	MAKİNA	11
PAZARLAMA VE PROGRAMLAR	MEKATRONİK	1
PAZARLAMA VE PROGRAMLAR	METALURJİ	1
PLANLAMA VE ENVANTER YÖNETİMİ	ENDÜSTRİ	19
PLANLAMA VE ENVANTER YÖNETİMİ	KİMYA	1
PLANLAMA VE ENVANTER YÖNETİMİ	MAKİNA	1
TEDARİK ZİNCİRİ YÖNETİMİ	ELEKTRİK – ELEKTRONİK	3
TEDARİK ZİNCİRİ YÖNETİMİ	ELEKTRİK	1
TEDARİK ZİNCİRİ YÖNETİMİ	ENDÜSTRİ	13
TEDARİK ZİNCİRİ YÖNETİMİ	KİMYA	1
TEDARİK ZİNCİRİ YÖNETİMİ	MAKİNA	16
TEDARİK ZİNCİRİ YÖNETİMİ	METALURJİ	3
TEDARİK ZİNCİRİ YÖNETİMİ	METALURJİ VE MALZEME	5
TESİSLER VE BAKIM	ELEKTRİK	1
TESİSLER VE BAKIM	ELEKTRONİK VE HABERLEŞME	1
TESİSLER VE BAKIM	ENDÜSTRİ	5
TESİSLER VE BAKIM	İMALAT	1
TESİSLER VE BAKIM	MAKİNA	8
ÜRETİM MÜHENDİSLİĞİ	ÇEVRE	1
ÜRETİM MÜHENDİSLİĞİ	ELEKTRİK – ELEKTRONİK	1
ÜRETİM MÜHENDİSLİĞİ	KİMYA	2
ÜRETİM MÜHENDİSLİĞİ	MAKİNA	16
ÜRETİM MÜHENDİSLİĞİ	METALURJİ	2
ÜRETİM MÜHENDİSLİĞİ	ÜRETİM	1
ÜRETİM VE PLANLAMA	ENDÜSTRİ	7
ÜRETİM VE PLANLAMA	MAKİNA	1
ÜRÜN DESTEK	ELEKTRİK ELEKTRONİK	1
ÜRÜN DESTEK	GIDA	1
ÜRÜN DESTEK	HAVACILIK	1
ÜRÜN DESTEK	MAKİNA	6

Figure 7 Number of Employees in Each Department by Engineering Field [3]

```python
import requests as requests

print("Input the URL:")
response = requests.get(input())
print()
if response:
    print(response.json().get("content", "Invalid quote resource!"))
else:
    print("Invalid quote resource!")
```

Figure 8 First Project Stage 1 Code

```python
from pprint import pprint
import requests as requests
from bs4 import BeautifulSoup

print("Input the URL:")
url = input()
response = requests.get(url, headers={'Accept-Language': 'en-US,en;q=0.5'})
if response:
    soup = BeautifulSoup(response.content, 'html.parser')
    p1 = soup.find("h1")
    p2 = soup.find("span", {'data-testid': 'plot-l'})
    if p1 is None or p2 is None:
        print("Invalid movie page!")
    else:
        movie = {"title": p1.text, "description": p2.text}
        pprint(movie)
else:
    print("Invalid movie page!")
```

Figure 9 First Project Stage 2 Code

```python
import requests as requests

print("Input the URL:")
url = input()
response = requests.get(url, headers={'Accept-Language': 'en-US,en;q=0.5'})
if response:
    with open("source.html", "wb") as f:
        f.write(response.content)
        print("Content saved.")
else:
    print(f"The URL returned {response.status_code}!")
```

Figure 10 First Project Stage 3 Code

```python
import string
import requests as requests
from bs4 import BeautifulSoup

base_url = "https://www.nature.com"
start_url = "/nature/articles?sort=PubDate&year=2020&page=3"
response = requests.get(base_url + start_url, headers={'Accept-Language': 'en-US,en;q=0.5'})
if response:
    soup = BeautifulSoup(response.content, 'html.parser')
    articles = soup.find_all("article")
    for article in articles:
        article_type = article.find("span", {'data-test': 'article.type'})
        if article_type.text.strip() == "News":
            article_link = article.find("a", {'data-track-action': 'view article'})
            article_response = requests.get(base_url + article_link.get("href"))
            article_soup = BeautifulSoup(article_response.content, 'html.parser')
            article_title = article_soup.find("h1").text.strip()
            article_body = article_soup.find("div", {"class": "c-article-body"}).text.strip().encode("utf-8")
            for i in (string.punctuation + '–' + '’'):
                if i in article_title:
                    article_title = article_title.replace(i, "")
            article_title = article_title.replace(" ", "_")
            print(article_title)
            with open(article_title + ".txt", "wb") as f:
                f.write(article_body)
else:
    print("Invalid page!")
```

Figure 11 First Project Stage 4 Code

```python
import os
import string

import requests as requests
from bs4 import BeautifulSoup

page_nums = int(input()) + 1
title = input()
base_url = "https://www.nature.com"
for page_num in range(1, page_nums):
    start_url = f"/nature/articles?sort=PubDate&year=2020&page={page_num}"
    response = requests.get(base_url + start_url, headers={'Accept-Language': 'en-US,en;q=0.5'})
    if response:
        directory_path = f"Page_{page_num}"
        os.mkdir(directory_path)
        soup = BeautifulSoup(response.content, 'html.parser')
        articles = soup.find_all("article")
        for article in articles:
            article_type = article.find("span", {'data-test': 'article.type'})
            if article_type.text.strip() == title:
                article_link = article.find("a", {'data-track-action': 'view article'})
                article_response = requests.get(base_url + article_link.get("href"))
                article_soup = BeautifulSoup(article_response.content, 'html.parser')
                article_title = article_soup.find("h1").text.strip()
                article_body = article_soup.find("div", {"class": "c-article-body"}).text.strip().encode("utf-8")
                for i in (string.punctuation + '–' + '’'):
                    if i in article_title:
                        article_title = article_title.replace(i, "")
                article_title = article_title.replace(" ", "_")
                with open(directory_path + "/" + article_title + ".txt", "wb") as f:
                    f.write(article_body)
```

Figure 12 First Project Stage 5 Code

```python
from nltk.tokenize import word_tokenize
from lxml import etree
from collections import Counter

xml_path = "news.xml"
news_cast = etree.parse(xml_path).getroot()[0]
for news in news_cast:
    print(news[0].text + ":")
    tokenized_text = word_tokenize(news[1].text.lower())
    freq_dict = Counter(sorted(tokenized_text, reverse=True))
    for el in freq_dict.most_common(5):
        print(el[0], end=" ")
    print()
    print()
```

Figure 13 Second Project Stage 1 Code

```python
import itertools
from string import punctuation
from nltk.tokenize import word_tokenize
from lxml import etree
from collections import Counter
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords

def lemmatize(token_list):
    lemmatizer = WordNetLemmatizer()
    lemmatized_list = []
    not_allowed_list = list(itertools.chain(punctuation, stopwords.words("english")))
    for token in token_list:
        token = lemmatizer.lemmatize(token)
        if token not in not_allowed_list:
            lemmatized_list.append(token)
    return lemmatized_list

xml_path = "news.xml"
news_cast = etree.parse(xml_path).getroot()[0]
for news in news_cast:
    print(news[0].text + ":")
    tokenized_text = word_tokenize(news[1].text.lower())
    tokenized_text = lemmatize(tokenized_text)
    freq_dict = Counter(sorted(tokenized_text, reverse=True))
    for el in freq_dict.most_common(5):
        print(el[0], end=" ")
    print()
    print()
```

Figure 14 Second Project Stage 2 Code

```
def purify_tokens(token_list):
    lemmatizer = WordNetLemmatizer()
    purified_list = []
    not_allowed_list = list(itertools.chain(punctuation, stopwords.words("english")))
    for token in token_list:
        token = lemmatizer.lemmatize(token)
        if token not in not_allowed_list and nltk.pos_tag([token])[0][1] == "NN":
            purified_list.append(token)
    return purified_list
```

Figure 15 Second Project Stage 3 Improved Version of the Lemmatization Method

```
def calculate_tf_idf(data_set):
    vectorizer = TfidfVectorizer(input='content', use_idf=True,
analyzer='word', stop_words=None, vocabulary=None)
    tfidf_matrix = vectorizer.fit_transform(data_set).toarray()
    terms = vectorizer.get_feature_names_out()
    word_dict = {}

    for i in range(len(data_set)):
        word_dict[i] = []
        for j, tfidf in enumerate(tfidf_matrix[i]):
            if tfidf != 0:
                word_dict[i].append((terms[j], tfidf))
        word_dict[i] = sorted(word_dict[i], key=lambda x: (x[1],
x[0]), reverse=True)
    return word_dict

xml_path = "news.xml"
news_cast = etree.parse(xml_path).getroot()[0]
titles = []
contents = []
for news in news_cast:
    titles.append(news[0].text + ":")
    tokenized_text = word_tokenize(news[1].text.lower())
    tokenized_text = sorted(purify_tokens(tokenized_text),
reverse=True)
    contents.append(' '.join(tokenized_text))
content_dict = calculate_tf_idf(contents)
for i in range(len(titles)):
    most_imp_counter = 0
    print(titles[i])
    while most_imp_counter < 5:
        print(content_dict[i][most_imp_counter][0], end=" ")
        most_imp_counter += 1
    print()
    print()
```

Figure 16 Second Project Stage 4 Changed Parts of the Code

```
import math

from bs4 import BeautifulSoup
from nltk.tokenize import sent_tokenize

file = open("news.xml", "r")
soup = BeautifulSoup(file, "xml")
news = soup.find_all("news")
for report in news:
    head, text = report.find_all("value")
    lines = sent_tokenize(text.text)
    imp_lines_num = round(math.sqrt(len(lines)))

    print("HEADER:", head.text)
    print("TEXT: ", end="")
    for i in range(imp_lines_num):
        print(lines[i].strip())
    print()
```

Figure 17 Third Project Stage 1 Code

```
import math
import statistics
import string

from bs4 import BeautifulSoup
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

meaningless_words = stopwords.words('english')
meaningless_words.extend(string.punctuation)
lemmatizer = WordNetLemmatizer()
file = open("news.xml", "r")

# Scraping Xml
soup = BeautifulSoup(file, "xml")
news = soup.find_all("news")


def find_most_prob_sent(mean_sents, lemm_sents):
    global word_stats

    # sort word_stats
    word_stats = {k: v for k, v in sorted(word_stats.items(),
key=lambda item: item[1], reverse=True)}

    # above dict creation does not preserve initial order because of
reverse
    most_prob_word = next(iter(word_stats))

    for k, v in mean_sents.items():
        sent_index = k
        if most_prob_word in lemm_sents[sent_index]:
            for lemm_word in lemm_sents[sent_index]:
                word_stats[lemm_word] = word_stats[lemm_word] *
word_stats[lemm_word]
            sent_ids.append(sent_index)
            return word_stats


def find_most_prob_sentences(lemm_sents):
    global word_stats

    # Calculating statistics
    stat_sents = []
    mean_sents = {}
    for i, lemm_sent in enumerate(lemm_sents):
        stat_sent = [word_stats[x] for x in lemm_sent]
        stat_sents.append(stat_sent)
        mean_sents[i] = statistics.mean(stat_sent)

    # sort mean prob of sentences
    mean_sents = {k: v for k, v in sorted(mean_sents.items(),
key=lambda item: item[1], reverse=True)}

    # Finding the most probable sentence
    word_stats = find_most_prob_sent(mean_sents, lemm_sents)
```

Figure 18 Third Project Stage 2 Code Part 1

```
for report in news:
    head, text = report.find_all("value")
    sentences = sent_tokenize(text.text)

    imp_lines_num = round(math.sqrt(len(sentences)))
    sent_ids = []
    sent_cnt = 0
    word_stats = {}

    # Tokenization
    tok_sents = []
    for sentence in sentences:
        tok_sents.append([x.lower() for x in word_tokenize(sentence)
if x.lower() not in meaningless_words])

    # Lemmatization
    lemm_sents = []
    lemm_sents_len = 0
    for tok_sent in tok_sents:
        lemm_sent = []
        for tok in tok_sent:
            lemmatized_tok = lemmatizer.lemmatize(tok)
            if word_stats.get(lemmatized_tok) is None:
                word_stats[lemmatized_tok] = 1
            else:
                word_stats[lemmatized_tok] += 1
            lemm_sent.append(lemmatized_tok)
        lemm_sents_len += len(lemm_sent)
        lemm_sents.append(lemm_sent)

    # Calculating Word Probabilities
    for key in word_stats.keys():
        word_stats[key] = word_stats[key] / lemm_sents_len

    while sent_cnt < imp_lines_num:
        find_most_prob_sentences(lemm_sents)
        sent_cnt += 1

    sent_ids.sort()
    text = [sentences[i] for i in sent_ids]
    print("HEADER:", head.text)
    print("TEXT: ", end="")
    print("\n".join(text))
    print()
```

Figure 19 Third Project Stage 2 Code Part 2

```
    # Tf-IDF approach
    sent_vectors = model.fit_transform([" ".join(sentence) for
sentence in lemm_sents]).toarray()
    sent_avg_vectors = [np.mean([e for e in vector if e != 0.]) for
vector in sent_vectors]
    sent_sorted = [sentence for _, sentence in
sorted(zip(sent_avg_vectors, lemm_sents), reverse=True)]

    sent_idx = [lemm_sents.index(sentence) for sentence in
sent_sorted[:imp_lines_num]]
    sent_idx.sort()

    text = [sentences[i] for i in sent_idx]
    print("HEADER:", head.text)
    print("TEXT: ", end="")
    print("\n".join(text))
    print()
```

Figure 20 Third Project Stage 3 Modified Part of the Code

```
    # Tf-IDF approach
    sent_vectors = model.fit_transform([" ".join(sentence) for
sentence in lemm_sents]).toarray()

    for word in head_sent[0]:
        index = model.vocabulary_.get(word)
        if index is not None:
            for vector in sent_vectors:
                if vector[index] != 0:
                    vector[index] *= extra_weight

    sent_avg_vectors = [np.mean([e for e in vector if e != 0.]) for
vector in sent_vectors]
    sent_sorted = [sentence for _, sentence in
sorted(zip(sent_avg_vectors, lemm_sents), reverse=True)]

    sent_idx = [lemm_sents.index(sentence) for sentence in
sent_sorted[:imp_lines_num]]
    sent_idx.sort()

    text = [sentences[i] for i in sent_idx]
    print("HEADER:", head.text)
    print("TEXT: ", end="")
    print("\n".join(text))
    print()
```

Figure 21 Third Project Stage 4 Modified Part of the Code

```python
from nltk.tokenize import WhitespaceTokenizer
from nltk.probability import FreqDist

with open(input(), "r", encoding="utf-8") as f:
    text = f.read()
    wst = WhitespaceTokenizer()
    tokenized_text = wst.tokenize(text)
    fdist = FreqDist(tokenized_text)
    print("Corpus statistics")
    print("All tokens:", fdist.N())
    print("Unique tokens:", fdist.B())
    print()
    while True:
        try:
            index = input()
            if index == "exit":
                break
            print(tokenized_text[int(index)])
        except IndexError:
            print("Index Error. Please input an integer that is in the range of the corpus.")
        except ValueError:
            print("Type Error. Please input an integer.")
```

Figure 22 Fourth Project Stage 1 Code

```python
from nltk.tokenize import WhitespaceTokenizer
from nltk.util import bigrams

with open(input(), "r", encoding="utf-8") as f:
    text = f.read()
    wst = WhitespaceTokenizer()
    tokenized_text = wst.tokenize(text)
    bigrams_text = list(bigrams(tokenized_text))
    print("Number of bigrams:", len(bigrams_text))
    print()
    while True:
        try:
            index = input()
            if index == "exit":
                break
            index = int(index)
            print(f"Head: {bigrams_text[index][0]}     Tail: {bigrams_text[index][1]}")
        except IndexError:
            print("Index Error. Please input an integer that is in the range of the corpus.")
        except ValueError:
            print("Type Error. Please input an integer.")
```

Figure 23 Fourth Project Stage 2 Code

```python
from nltk.tokenize import WhitespaceTokenizer
from nltk.util import bigrams

with open(input(), "r", encoding="utf-8") as f:
    text = f.read()
    wst = WhitespaceTokenizer()
    tokenized_text = wst.tokenize(text)
    bigrams_text = list(bigrams(tokenized_text))
    model = {}

    for head, tail in bigrams_text:
        if model.get(head) is None:
            model[head] = {}
            model[head][tail] = 1
        else:
            if tail in model[head]:
                model[head][tail] += 1
            else:
                model[head][tail] = 1
    while True:
        try:
            print()
            head = input()
            if head == "exit":
                break
            print(f"Head: {head}")

            counter = 0
            for tail, cnt in sorted(model[head].items(), key=lambda item: item[1], reverse=True):
                print(f"Tail: {tail}    Count: {cnt}")
                if counter >= 2:
                    break
                counter += 1
        except IndexError:
            print("Index Error. Please input an integer that is in the range of the corpus.")
        except KeyError:
            print("Key Error. The requested word is not in the model. Please input another word.")
```

Figure 24 Fourth Project Stage 3 Code

```python
    sentence_cnt = 10
    word_cnt = 9  # One of them is predetermined
    head = random.choice(list(model.keys()))
    sentence = head + " "
    for i in range(sentence_cnt):
        for j in range(word_cnt):
            head = random.choices(list(model[head].keys()), list(model[head].values()))[0]
            sentence += head + " "
        print(sentence.strip())
        sentence = head + " "
```

Figure 25 Fourth Project Stage 4 Modified Part of the Code

```
sentence_cnt = 10
sentence = []
for i in range(sentence_cnt):
    word_cnt = 0
    while True:
        word = random.choice(list(model.keys()))
        if re.match("^[A-Z].*[^.!?]$", str(word)):
            word_cnt += 1
            sentence.append(word)
            break
    while True:
        word = random.choices(list(model[word].keys()), list(model[word].values()))[0]
        sentence.append(word)
        word_cnt += 1
        if word_cnt >= 5:
            if re.match(".*[.!?]$", word):
                break
    print(" ".join(sentence))
    sentence.clear()
```

Figure 26 Fourth Project Stage 5 Modified Part of the Code

```
import random
import re
from nltk.tokenize import WhitespaceTokenizer
from nltk.util import trigrams

with open(input(), "r", encoding="utf-8") as f:
    text = f.read()
    wst = WhitespaceTokenizer()
    tokenized_text = wst.tokenize(text)
    trigrams_text = list(trigrams(tokenized_text))
    model = {}
    for head_1, head_2, tail in trigrams_text:
        head = head_1 + " " + head_2
        if model.get(head) is None:
            model[head] = {}
            model[head][tail] = 1
        else:
            if tail in model[head]:
                model[head][tail] += 1
            else:
                model[head][tail] = 1

    sentence_cnt = 10
    sentence = []
    for i in range(sentence_cnt):...
```

Figure 26 Fourth Project Stage 6 Modified Part of the Code

```
import pandas as pd

scores = []
reviews = []

with open(input(), "r") as file:
    for line in file:
        review, score = line.rsplit(sep=",", maxsplit=1)
        scores.append(score.replace("\n", ""))
        reviews.append(review.strip())

info_tuple = list(zip(reviews[0:150000], scores[0:150000]))
film_infos = pd.DataFrame(info_tuple, columns=["Review", "Score"])

print(film_infos.head(5))
print(film_infos.tail(5))
```

Figure 27 Fifth Project Stage 1 Code

```
stopwords = stopwords.words("english")
lemmatizer = WordNetLemmatizer()


def lemmatize(word_list):
    lemmatized_list = []
    for word in word_list:
        if word not in stopwords:
            lemmatized_word = lemmatizer.lemmatize(word)
            lemmatized_list.append(lemmatized_word)
    return lemmatized_list


# Tokenization
film_infos["Lemmas"] = film_infos["Review"].apply(
    lambda x: [y.lower() for y in regexp_tokenize(x, "[A-Za-z]+")])

# Lemmatization
film_infos["Lemmas"] = film_infos["Lemmas"].apply(
    lambda x: lemmatize(x))

print(film_infos.head(5)["Lemmas"])
print(film_infos.tail(5)["Lemmas"])
```

Figure 28 Fifth Project Stage 2 Added Part of the Code

```
with open("positive_words.txt", "r") as file:
    pos_words = [x.replace("\n", "") for x in file.readlines()]

with open("negative_words.txt", "r") as file:
    neg_words = [x.replace("\n", "") for x in file.readlines()]

with open(input(), "r") as file:
    for line in file:
        review, score = line.rsplit(sep=",", maxsplit=1)
        scores.append(score.replace("\n", ""))
        reviews.append(review.strip())

# Sentiment Analysis
sent_anal_res = pd.Series([0, 0, 0])
sent_anal_labs = pd.Series(["Negative", "Neutral", "Positive"])
for film_info in film_infos["Lemmas"]:
    score = 0
    s = pd.Series(film_info)
    score += s.isin(pos_words).sum()
    score -= s.isin(neg_words).sum()
    if -3 <= score <= 3:
        sent_anal_res[1] += 1
    elif score < -3:
        sent_anal_res[0] += 1
    else:
        sent_anal_res[2] += 1
plt.pie(sent_anal_res, labels=sent_anal_labs, autopct="%d%%")
plt.show()

total_reviews = sent_anal_res.sum()
print(f"""The answer to the 1st question: {sent_anal_labs[sent_anal_res.idxmax()]}
The answer to the 2nd question: {round(sent_anal_res.min() / total_reviews * 100)}%
The answer to the 3rd question: {round((sent_anal_res[2] - sent_anal_res[0]) / total_reviews * 100)}%""")
```

Figure 29 Fifth Project Stage 3 Added Part of the Code

```
def label_sentiment(score):
    score = int(score)
    if 7 <= score <= 10:
        return "Positive"
    elif 1 <= score <= 4:
        return "Negative"


# Sentiment Analysis
film_infos["Sentiments"] = film_infos["Score"].map(lambda x: label_sentiment(x))

flattened_lemmas = []
for film_info in film_infos["Lemmas"]:
    flattened_lemmas.append(" ".join(film_info))

vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(flattened_lemmas)
x_train, x_test = train_test_split(tfidf_matrix, train_size=0.75, random_state=42)

print(f"""The size of the TF-IDF dictionary is {tfidf_matrix.shape[1]} words.
The size of the training set is {x_train.shape[0]} rows.
The shape of the matrix of test data is {x_test.shape}.""")
```

Figure 30 Fifth Project Stage 4 Modified Part of the Code

```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import classification_report


def sentiment_process(sc):
    if sc > 5:
        return 1
    return 0


input_ = input()
path = "data.pkl"
data = pd.read_pickle(path)

data["sentiment"] = data.Score.map(sentiment_process)

X = data.Lemmas.map(lambda x: " ".join(x))
y = data.sentiment
x_train, x_test, y_train, y_test = train_test_split(X, y, train_size=0.75, random_state=42)

tfidf = TfidfVectorizer(input=x_train.tolist(), ngram_range=(1, 3))

x_train_tr = tfidf.fit_transform(x_train)
x_test_tr = tfidf.transform(x_test)

clf = SGDClassifier(alpha=5e-6, fit_intercept=True, learning_rate='optimal', loss="hinge", penalty='l2')
clf.fit(x_train_tr, y_train)

y_pred = clf.predict(x_test_tr)

print(classification_report(y_test, y_pred))
```

Figure 31 Fifth Project Stage 5 Optimized Final Code

Self-Checklist for Your Report

Please check the items here before submitting your report. This signed checklist should be the final page of your report.

- ☑ Did you provide detailed information about the work you did?
- ☑ Is supervisor information included?
- ☑ Did you use the Report Template to prepare your report, so that it has a cover page, has all sections and subsections specified in the Table of Contents, and uses the required section names?
- ☑ Did you follow the style guidelines?
- ☑ Does your report look professionally written?
- ☑ Does your report include all necessary References, and proper citations to them in the body?
- ☑ Did you remove all explanations from the Report Template, which are marked with yellow color? Did you modify all text marked with green according to your case?

Signature: