İZMİR INSTITUTE OF TECHNOLOGY

ELECTRICAL AND ELECTRONICS ENGINEERING DEPARTMENT

SUMMER PRACTICE REPORT

Name - Last Name : Emre SENGİR

Turkish Identification # : **43936669590**

Student # : **270206023**

Company Name : **IYTE Departmant of Computer Engineering**

Dates : 10/07/2023 - 18/08/2023

Summer Practice # : **EE300**

İZMİR

T.C.
İZMİR INSTITUTE OF TECHNOLOGY
FACULTY OF ENGINEERING
ELECTRICAL & ELECTRONICS ENGINEERING DEPARTMENT
SUMMER PRACTICE REPORT INNER COVER

Name – Last Name : Emre SENGİR

T.R. Identification # : 43936669590

Student # : 270206023

COMPANY / FIRM

Name : IYTE Faculty of Computer Engineering

Address : Gülbahçe, Urla İzmir 35430, Türkiye 0 232 750 7860

Phone : 0 232 750 7860

Fax : -

E-mail : -

Starting Date : 10 July 2023

Ending Date : 18 August 2023

SUPERVISOR AT COMPANY

Name – Last Name : Buket ERŞAHİN

Title : Instructor

Contact Info : buketoksuzoglu@iyte.edu.tr

Signature and Stamp :

1. We encourage our students to start writing internship reports during their internships. In this way, they will have the chance to do an internship in awareness of what is required of them.

2. Internship reports must be prepared in accordance with the regulations defined in this document. **Reports that do not comply with these conditions are not assessed and the internship of the student is considered to be unsuccessful.**

3. The reports must be in English and written with computer with **your own words**. Drawing should conform to acceptable engineering standards). The report must not exceed **10 pages in total**.

4. When sources or documents are used in the report from other resources such as internet, company sources, books, data sheets etc., they should be specified both in the text where they are used and in the **References** section. The reports must not consist of cut and paste parts from other sources. The whole report must written in student's own words. In mandatory cases, the tables and figures can be copied, but still must be cited in the text and referenced in the **References** section of the report. The students are responsible for knowing the contents of their reports. When necessary, the students may be invited to the oral exam and respond to the questions about the content of their reports.

5. Internship reports should provide information indicating that engineering activities complementary to the education received at the department have been performed at the company.

6. In the internship reports, the name and contact info of the supervisor must be clearly indicated and the signature and the firm stamp must exist.

7. The reports should be prepared and printed on **A4** size white papers in **1,5 line** spacings in **justified** paragraphs using **12 pt Times New Roman** fonts, with the **top, bottom, and right on 2.2 cm**, and **left on 3 cm**. Main headings are centered and written in capital boldface. Subtitles should be written in small letters and boldface. Drawings should conform to acceptable engineering standards.

8. **Internship reports should be submitted in spiral bound or in filed form, and internship evaluation forms should be presented in closed envelops and approved form. Otherwise the reports will not be evaluated and the internship of the student will be considered as unsuccessful.**

EEE Department Internship Commission

I declare that I have prepared my internship report according to the regulations and notes above.

Student's Name and Last Name: Emre Sengir

Student's Signature: 

TABLE OF CONTENTS

1. DESCRIPTION OF THE COMPANY

The place that I am going to carry out my internship is Computer Engineering Faculty of the Izmir Institute of Technology. IYTE is located after Urla, where the crossroads between Urla, Karaburun and Çeşme there exist the Institution. IYTE mostly based on engineering and mathematical based faculties, one of them is the Computer Engineering faculty that educates it's student in mostly software design based lessons, so in order to educate myself in advance mathematical based software and improve my skillset I decided to do one of my internships in academic standards. During my internship I will be studying with one of the academic members of the department, Buket ERŞAHİN. Buket ERŞAHİN is responsible for her own departments internship situations and gives lectures on introduction to programming, software engineering and data mining which is a part of natural language processing.

2. INTRODUCTION

Main goal of this internship is to develop a project on a machine learning branch, Natural Language Processing, but as a starter I hold the purpose of gaining an insight into the general concept of machine learning and it's other branches. So while learning about the topic I am going to try to embody it in a productive way and conclude our research.

Name and description of the research project can be explained as Multi-Document Summarization in Turkish, which holds to purpose of shortening or extracting the necessary information on any kind of text while processing the text. Term 'Extracting' may create some confusion during the flow of project that will be explained further, so in order to explain ourself clearly, by extracting we mean that we want to take the important information of the text in some certain ways, but for our ultimate purpose we want to do it by creating AI made sentences by using deep learning models similar to the paraphrasing what we humans do. Extracting is also another tool that can be used like determining and selecting the sentences that has importance to use on learning model in order to obtain efficiency and neglecting noise or unnecessary items like words.

As we explained we tried to develop our model as advanced as possible, details about the flow of project will be given in detail on body of the report part.

3. BODY OF THE REPORT

In the beginning of the internship we first decided to create a roadmap to work systematically. Since I have no history of machine learning studies, we decided to leave two out of six weeks of total internship time to educational purposes about topic with courses and research papers that has been written previously about specified project on multi-document summarization, machine learning etc..

During these two weeks I have studied and practicalized on course [1] which introduces machine learning branches and methods in a general way and sketchy manner. There are many machine learning models that is fit for many types of datasets. Basic purpose of machine learning is to predict an outcome by evaluating previous data that is gathered on a dataset. In a short explanation, basic applications of machine learning models is most similar to the mathematics like a supervised, any numbers of input and existing output model, for example the relation between the amount of fertilization on a field and amount of harvested product from that field, can be formed as a linear mathematical function. As I stated this example is the most simple application of a machine learning model. There are types that evaluates not the outcome of dataset whether the patterns between the independent variables, called as unsupervised model, which is most likely to determine characteristics of the dataset and cluster or classify the multi independent variables to determine the outcome in supervised model.

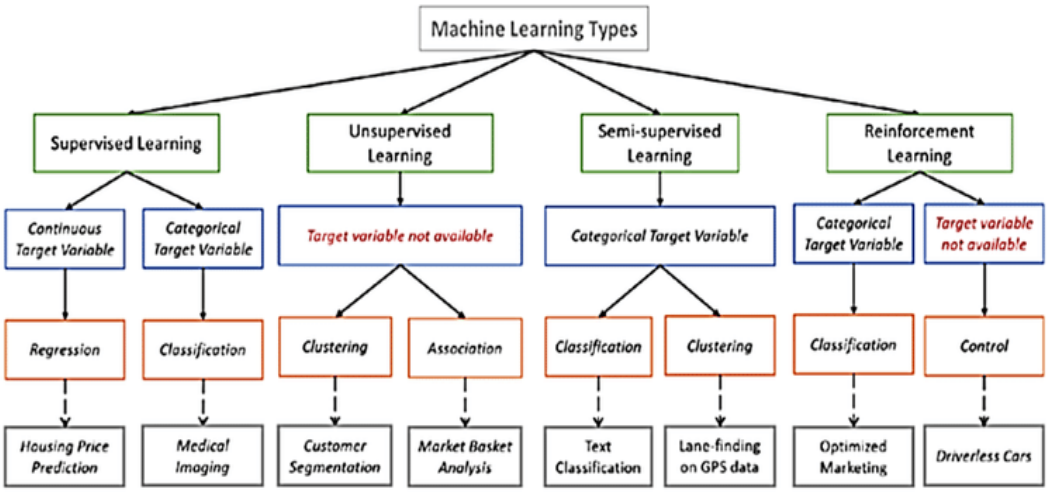

Chart 1: Machine learning models. [2]

As you can see from the above chart there are many types of machine learning models and they can be applied to evaluate dataset in different steps and reorganized product can again be evaluated to obtain the outcome.

In continuation to the course, in the following weeks I studied about the subject my Instructor gave me, I have made readings and researches on my Instructor forwarded to me about the previous studies of the work. As I stated afore our research topic requires more than a sketchy knowledge on natural language processing. Most of the common works, or basics, about natural language processing are sentimental analysis and sentence classification by categorization. So one can understand that it's not enough to build a model that can generate sentences and analyze previous data to evaluate it in a generative way. Our preliminary research steps on specified topic is as follows:

1) Text Preprocessing (cleaning text, stemming, any-type of removals, etc.).
2) Database management, proper sentence transformer applications and embedding.
3) Types of text summarization models.

Starting from the step one text preprocessing, on the previous studies there was an important point that ascends our study in a complicated level. This point was, as todays' most used language English, there are certain morphological differences between Turkish and English that needs to be analyzed on preprocessing step to clean text properly and evaluate it. I pointed out this because even though we study on Turkish sources, most of the core programming libraries includes English based language processing tools and creates inconsistency in Turkish. Before I explain how I solved it on the final of the project, I want to explain the problem properly. During the stemming operations words are passed into a function to transform their root or a certain number of letters length versions. In English this is not an important problem because most of the time in comparison to the Turkish words, sentences in English can hold their meaning most of the time after stemming, but in Turkish, knowing that it's an Agglutinative language, words can have more than one affixes, thus they can lose their identity after stemming, resulting in sentence to lose meaning.

Word	Analysis
gören *(the one who sees)*	gör+en(DB)
görülen *(the one which is seen)*	gör+ül(DB)+en(DB)
görüş *(opinion)*	gör+üş(DB)
görüşün *(your opinion)*	gör+üş(DB)+ün
görüşler *(opinions)*	gör+üş(DB)+ler
görüşme *(negotiation)*	gör+üş(DB)+me(DB)
görüşmelerin *(of negotiations)*	gör+üş(DB)+me(DB)+ler+in

Table 1: Analyzes of root word 'gör' with different affixes, derivational boundaries represented by (DB). [3]

This problem has been studied in a whole research during the implementation of text summarization by using a hybrid approach using both extraction and abstraction. Applying customized stemming policies that one like stemming with respect to certain level of derivational boundaries, researches evaluated the results with different evaluation tools and in result they observed that most successful stemming policies are Prefix10[1] and root stemming.[3] Prefix10: It is that words are stemmed to keep their 10 letters at max.

Previous studies has a lot points to explain on but in order to keep report short and whole I explained the most important ones for now and in further parts. After determining the most useful stemming policy (root stemming) my instructor recommended me to make examinations on VNLP library [4]. VNLP is a library for Turkish language processing that stores common functions to process words in Turkish like stemming analysis and stop-word removal etc.. I stated the function stemming analysis exactly because function is not use for to clean words of a sentence to their root form, rather to analyze what types of affixes that word have and the type of word. With a little correction on the source code I made it useful for stemming operation that takes a sentence as an input and returns lowered and root-stemmed versions of words as a list. But text preprocessing is not just stemming operation, and including that function returns a list, returned value required to be transformed into a string again after removing stop-words in the sentence.

```
sentence = [[], []]
for token in tokens:
    sentence[0].append(token)
    candidate_analyzes = self.candidate_generator.get_analysis_candidates(token)

    root_tags = []
    for analysis in candidate_analyzes:
        root = analysis[0]
        tags = analysis[2]
        if isinstance(tags, str):
            tags = [tags]
        joined_root_tags = "".join([root])
        root_tags.append(joined_root_tags)

    sentence[1].append(root_tags)
```

Code Snippet 1: Manipulated for loop that performs the stemming in the function StemmerAnalyzer.predict().

By manipulation I mean the correction in the code, it is so simple that in the original version of the code line "joined_root_tags = "".join**([**root**])**", takes tags=[tags] in addition to join([root]) so original version is "joined_root_tags = "".join**([**root+tags**])**" thus we handled the most important part of text preprocessing and applied stop-word removal and combining the list in type string again by simply,

```
def combine(sentence):
    whole = ''
    for i in sentence:
        whole += i+' '
    return whole

def stem_stop_remover(sentence):

    temp = stemmer.predict(str(sentence))
    final = stp_rmv.drop_stop_words(temp)
    return final
```

Code Snippet 2: Text preprocessing functions in module "handy_f" that is used for project.

After completing text preprocessing part in short I would like to continue on the subject that how am I supposed to perform text summarization. As I referred in study [3] researchers performed a hybrid approach for text summarization which consists extraction and abstraction. Extraction is a simple approach for text summarization, one can perform it determining one or many sentence ranking functions with respect to any characteristic of sentence like term frequencies or sentence location in text. In the other hand abstraction is more human-like approach which performs a similar operation what we humans do as

paraphrasing. In order to widen the scope of research I also decided to study on hybrid model, but before diving into detail I would like to express steps that I took during project:

1) Data gathering.
2) Embedding.
3) Classification.
4) Extraction.
5) Abstraction.

We decided that we can perform multi document summarization by classifying similar text according to a query of an user input and by adding the most related ones under one string to extract important informations then performing abstraction. So we can say that first three steps are connected to each other naturally. Our dataset is decided to be formed up from FAQ of THY [5], we can say that it will include three columns; topic, question, answer. All need to know about structure of dataset is their columns for now. Since questions are short sentences in comparison to their answers, in order to perform quick classification on query we embedded questions by using a simple library called chromadb [6]. Chromadb is setup library for English document database management with the help of embedding, but one can change the embedding function it uses by defining it another one to use. Embedding is simply transforming documents to vector representations to measure their mathematical distances for classification. Embedding has parameters like document itself, it's embedding (vector), metadata (topic of document) and ids (index). Metadata parameter is the reason we included a topic column to dataset, it resulted in increasing the performance on classification.

```
In [15]: collection.get(["0"])

Out[15]: {'ids': ['0'],
          'embeddings': None,
          'metadatas': [{'konu': 'Check-in'}],
          'documents': ['  check-in yap koltuk numara değiş ?']}
```

```
In [23]: a = collection.peek(1)

In [24]: a['embeddings'][0][0]

Out[24]: 0.0214811023324728

In [25]: len(a['embeddings'][0])

Out[25]: 768
```

Code Snippet 3: Embedded stemmed docs.

In order to perform proper embedding we used a Turkish sentence-transformer function [7]. Function transforms given document parameter to a 1x768 dimensional vector representation. As I stated afore collection represents chromadb client that performs the operations in the given code snippet above. In addition, in the document parameter of the given code snippet you can see an example of a stemmed sentence.

```
In [34]: sentence = comb(['uç', 'iptal', 'et', ',', 'iade', 'veya', 'başka', 'bir', 'uç', 'için', 'nasıl', 'bir', 'süreç', 'izle', 'gerek'
         model = SentenceTransformer('emrecan/bert-base-turkish-cased-mean-nli-stsb-tr')
         sample_embedding = model.encode(sentence)
         upd = sample_embedding.tolist()
         found = collection.query(query_embeddings = upd, n_results = 3)
         found

Out[34]: {'ids': [['43', '41', '29']],
          'distances': [[277.0673828125, 291.42291259765625, 294.94287109375]],
          'metadatas': [[{'konu': 'Uçuş iptali ve değişikliği'},
            {'konu': 'Uçuş iptali ve değişikliği'},
            {'konu': 'Ücret koşulları'}]],
          'embeddings': None,
          'documents': [['  uç iptal et başka bir uç yön ?',
            '  uç iptal et bilet ücret iade et ?',
            '  bilet iptal et , iade al ?']]}
```

Code Snippet 4: Query.

To perform query on the given input, assuming input has been stemmed as you can see from above, “sentence = comb([])” comb is a reference to the function that combines list of stemmed words ,which is the output of vnlp stemmer analyzer, to a string. I performed the following above, stored dictionary in found parameter and used ‘ids’ to get the answers indices.

After I get the indices and draw out the related answers in under one string, now comes the summarization part. As I stated above I decided to perform a hybrid approach and first thing to do was extraction in order to filter important sentences by sentence ranking functions.

On one of the studies I examined there was a similar model to the way perceptron works in a neural network which has two properties in basics; activation function and weight. On the study, researches determined several sentence scoring functions and assigned coefficients (weights) to them in order to obtain final score of the sentence [8]. Due to the short time interval of the internship I am unable to perform such detailed extraction operation instead I performed basic extraction by using TF-IDF method. TF-IDF is abbreviation for term frequency and inverse document frequency, term frequency determines the key words in a text and measures the score of sentence by looking how many it holds of these words. Inverse document frequency on the other hand determines rarity of a sentence by looking how many unique words in the text that sentence holds and evaluates

accordingly. By creating matrices of the scored sentences accordingly to TF-IDF after performing matrix multiplication in between them I obtained resultant matrix of scored sentences and halvened the overall sentences, and gathered the rest under another string.

In the final step we are supposed to perform an abstraction of the extracted text. Advanced natural language processing methods uses neural networks most of the time in order to generate sentences. In difference to the commonly used neural networks like Artificial and Convolutional that perform unidirectional operation, except the feedback mechanism to rearrange weights, in order to generate a text we need a repetitive model, likely to what we humans do while creating meaningful sentences knowing the structure of the sentence or relating between words to protect meaning.

Recurrent neural networks is similar to the convolutional neural networks but as I stated it performs the operation by taking another certain properties. For example in a word based generation after the neural network is trained and an input is given, output words constantly fed to the next perceptron and passed onto some gates/functions to calculate certain values. What a recurrent neural network does is similar to the flowchart given below.

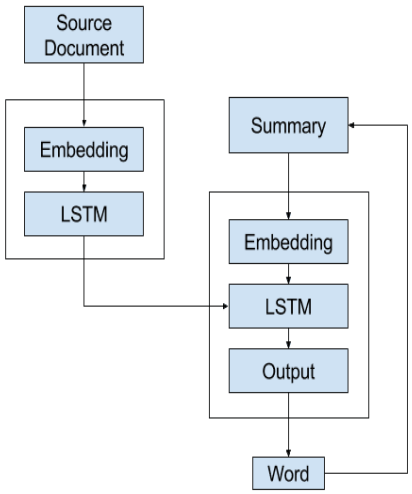

Chart 2: Approximate flowchart of what a recurrent neural network does. [9]

Thus it creates a sequential model which generates sentences word by word. In order to keep the operation simple through many advanced models I needed to choose one that I can modify and implement in time. One of the model that I catch interest on was using a basic approach, by using Plato's Republic book as a text input. After preprocessing the text it creates sequential list of words that moves word by word along the text in a determined length like 10+1, which makes eleven words of list and next list will remove the first word

of the list and include the next word in the text. Thus moving word by word and transforming them into 10 elements of input layer and 1 element of output layer writer trains the dataset with a corpus size of word_count_of_text+11.

```
In [17]: print("",sequences[1],"\n",sequences[2],"\n",sequences[3])

         tarihinden öncesinde alınmış bir biletim var ücretsiz bagaj hakkım ne kadar
         öncesinde alınmış bir biletim var ücretsiz bagaj hakkım ne kadar olur
         alınmış bir biletim var ücretsiz bagaj hakkım ne kadar olur mayıs
```

Code Snippet 5: Slipping in sequences.

After forming our corpus we separate the last words of sentences on another matrix and categorized (or vectorizing similar to bag of words) them for fitting into neural network, also embedding the input layer to fit. I trained our extracted sequences of sentences then giving the user input as seed text I performed text generation in certain number of words as output to the user input which is answer to the question.

Except the final week I researched, studied and tested analysis on dataset separately in each step. In the final week of the internship I gathered all the steps as several functions under one module named handy_f and used it in main_project code file to perform and show the steps clearly. When I finished to implement codes, two main problems occurred when I tried to execute and perform multi-document summarization. One is that performing stemming and stop-word removal surely simplifies text but in embedding step of recurrent neural network when we are supposed to obtain list of sequences having dimensions as (word_count+11)x11, due to the punctuation marks that have not been removed from the text, different sizes of sequences showed up. Second one is that the common forwarding sentences on answers of questions like “detaylı bilgi için x.com adresine gidiniz” is likely to be occur in many answers we gather before extraction, and during extraction these sentences gained importance with term frequency scoring and created unnecessary noise by showing up many times after extraction supplying no information to the summary.

In addition we can say that punctuation and common sentences between texts removal, are steps to be considered to improve and perform the code. Nonetheless I wrote a research paper about the work done with its good and bad, proposed alternative solutions to the previous studies and new tools, libraries etc., paper is also a demonstration of work done during internship.

5. CONCLUSION

Main reason that I intended to perform internship on such a topic that is not very related to our lectures, is that growing potential in machine learning and artificial intelligence, also to practice my skill in another language like python.

In conclusion I ended up learning machine learning methods in a general approach. I worked with different types of datasets to apply different types of learning models while there are several models for certain types of datasets and purposes, I can evaluate the performance of the specific model and arrange some accuracy increasing additions or extractions on datasets for the necessary situations. As I stated in introduction part this took my two weeks to complete and what have I learned was not enough to perform the project requirements, thus I made searching for extra studies and practices with guidance from my Instructor to obtain what is required to perform the project.

The most important thing that I have learned after spending my third week on mostly searching and studying on natural language processing is that if I have spend my time mostly on coding or practicing I may have completed the project in a more advanced result. It may ended up in reverse effect though, one can not know before try. Even though some mistakes that I had done during the planning of internship, during the internship, I think I have achieved text mining in certain level, and gained insights about the deep learning models for further researches. The two error I encountered in final week detailly explained in our research paper. We decided that they can be adjusted and we may finish the project in near future.

To sum up, from this internship I learned many things that I have no knowledge about before the internship. I can form datasets by using certain tools like web-scrapping, process these datasets to make them useful for learning models, make evaluations and adjustments on dataset and learning model to obtain better accuracy while preventing it to reach overfitting or underfitting cases, considering dataset is made up from texts like our research I can perform text mining and language processing etc..

REFERENCES

[1] https://www.udemy.com/course/machinelearning/

[2] Sengupta, Rishiraj & Sengupta, Dhritiraj & Kamra, Aashish & Pandey, Digvijay. (2020). JOURNAL OF CRITICAL REVIEWS ARTIFICIAL INTELLIGENCE AND QUANTUM COMPUTING FOR A SMARTER WIRELESS NETWORK. 10.31838/jcr.07.19.21.

[3] Nuzumlalı, Muhammed & Ozgur, Arzucan. (2014). Analyzing Stemming Approaches for Turkish Multi-Document Summarization. 10.3115/v1/D14-1077.

[4] https://vnlp.readthedocs.io/en/latest/

[5] https://www.turkishairlines.com/tr-int/bilgi-edin/index.html

[6] https://docs.trychroma.com/api-reference

[7] https://huggingface.co/emrecan/bert-base-turkish-cased-mean-nli-stsb-tr#citing--authors

[8] Mücahid Kutlu and others, Generic Text Summarization for Turkish, *The Computer Journal*, Volume 53, Issue 8, October 2010, Pages 1315–1323, https://doi.org/10.1093/comjnl/bxp124

[9] https://machinelearningmastery.com/encoder-decoder-models-text-summarization-keras/, and his book,

Brownlee, Jason, 2017, “Chapter 20: Project: Develop a Neural Language Model for Text Generation”, Deep Learning for Natural Language Processing: Language Modelling, Sarah Martin, pg. 226-245.