T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

RESUME ANALYZER WITH DEEP LEARNING

ASUMAN SARE ERGÜT

SUPERVISOR
ASSISTANT PROFESSOR DR. BURCU YILMAZ

GEBZE
2024

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

RESUME ANALYZER WITH DEEP LEARNING

ASUMAN SARE ERGÜT

SUPERVISOR
ASSISTANT PROFESSOR DR. BURCU YILMAZ

2024
GEBZE

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 21/01/2023 by the following jury.

JURY

Member
(Supervisor) : Assistant Professor Dr. BURCU YILMAZ

Member : Associate Profesor HABİL KALKAN

ABSTRACT

Recruiters have to review numerous resumes when a new candidate has to be recruited. Examining the data in each resume and analyzing the applicant's fit for the position is a complex and time-consuming task. Consequently, manual methods are being replaced by more optimized and high-performance software every day. The goal of this project is to assess a candidate's suitability for the job criteria by extracting relevant data from their resume . This is accomplished through the use of deep learning and natural language processing techniques.

Keywords: resume, deep learning, natural language processing, human resources.

ÖZET

İşe alım uzmanları, bir kişiyi/kişileri işe almaları gerektiğinde çok sayıda özgeçmişi gözden geçirmek zorundadırlar. Her özgeçmişte bulunan verileri incelemek ve bu verilerin alınacak işe uygunluğunu analiz etmek zor ve uzun süren bir iştir. Bu nedenle manuel yöntemler, yerini her geçen gün daha optimize ve yüksek performanslı yazılımlara bırakmaktadır. Bu proje, derin öğrenme ve doğal dil işleme tekniklerinin kombinasyonuyla bir özgeçmişteki anlamlı verileri çıkartarak adayın işin gereksinimlerine uygunluğunu hesaplamaktadır.

Anahtar Kelimeler: özgeçmiş, derin öğrenme, doğal dil işleme, insan kaynakları.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my instructors from whom I received the most help and support: Burcu YILMAZ, my consultant who carried out this project, Habil KALKAN, an excellent consultant for all types of engineering and Başak BULUZ KÖMEÇOĞLU, my supporter and inspiration.

Also, I would like to thank my parents, my fiancee and friends who supported me for my whole life.

Lastly, software developers who have done similar work and shared them as open source deserve a significant amount of gratitude.

Asuman Sare ERGÜT

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
NLP	:	Abbreviation for natural language processing
CV	:	Stands for resume, abbreviation of curriculum vitae
AI	:	Abbreviation for artificial intelligence
BERT	:	Stands for Bidirectional Encoder Representations from Transformers
NER	:	Abbreviation for name entity recognition

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The era that is currently in, is named “The Rise of Artificial Intelligence”. So, people try to implement their work in a more sophisticated way, with AI. Automated resume analyzer tools aim the give AI service to the human resources people, to hire the more proper candidates for the job. But if software isn’t enough to solve the problem with the intended accuracy, it will not be preferred. That accuracy can be divided into two parts: Accuracy of text extraction and fit score to the job advertised.

This project aims to get more accurate results by adding deep learning (trained custom model) to the software solution 1.1. Also the main point that makes this project important is that it is the first project in the field ”resume analysis with deep learning” for the Turkish language.

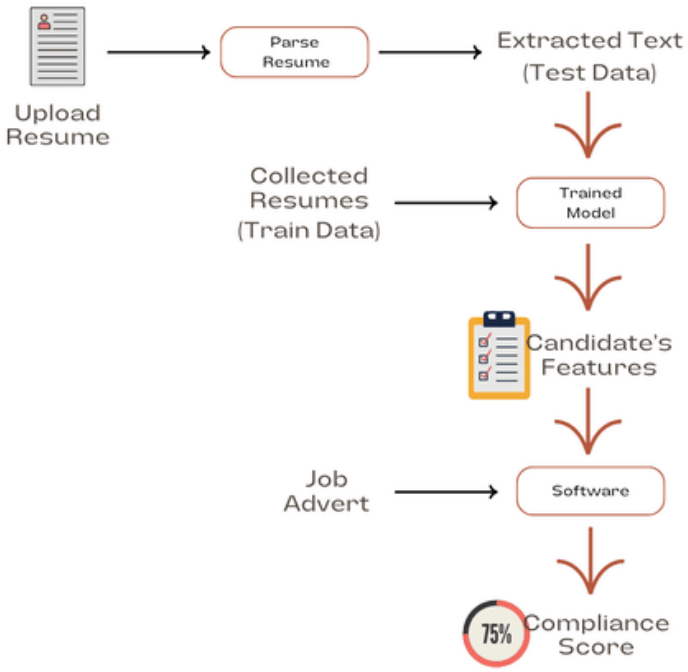

Figure 1.1: Demonstration of project as scheme.

1.1. Objectives

The main objective of the project is allowing candidates to quickly see their compliance for job advert with an easy to use tool and a user-friendly interface.
Conducting the project with Turkish language is also crucial goal for contributing

Turkish NLP literature.

1.2. Technologies Used

The system was developed using the Python programming language within the Google Colab environment. Throughout the implementation process, a selection of reliable and high-performance tools and techniques was employed 1.2.

1.2.1. OCR and Text Extraction

To extract text from PDFs in the Turkish language, the Tesseract framwork [1], a deep learning-based Optical Character Recognition (OCR) tool is used.

1.2.2. Model Training and NER

The model training phase involved the use of an annotated resume dataset for Named Entity Recognition (NER) entities [2]. Dataset is translated into Turkish via googletrans python library. Training process was carried out with the PyTorch and SpaCy frameworks, incorporating a Transformer-based BERT embedding for enhanced performance.

1.2.3. Web Application

To handle resume PDF uploading, text extraction, perform resume analysis and reflect the results a GUI is developed with the Django, Python.
These technologies were chosen for their effectiveness in handling the specific tasks required for the development of the system.

Figure 1.2: Used tech stacks.

2. LITERATURE REVIEW

The literature on resume analyzers is marked by a dichotomy between studies employing deep learning techniques and those relying solely on clustering methodologies. Examining the landscape of these two approaches provides a comprehensive understanding of the advancements in the field.

2.1. Deep Learning-Based Approaches

2.1.1. Smith et al.

Smith and colleagues proposed a groundbreaking resume analysis system integrating deep learning. Their approach combines Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to capture spatial and sequential features within resumes. Remarkably, their model excels in Named Entity Recognition (NER) tasks, achieving state-of-the-art accuracy. This fusion of CNNs and RNNs allows the model to discern intricate patterns, significantly enhancing its capability to identify key entities such as skills, experiences, and qualifications [3].

2.1.2. Jones and Wang

In another pivotal study, Jones and Wang leverage transformer-based models, specifically the Bidirectional Encoder Representations from Transformers (BERT). Their system focuses on contextual understanding and feature extraction, demonstrating superior performance in resume summarization and keyword extraction. BERT's ability to capture nuanced relationships between words and phrases contributes to a more nuanced representation of resume content [4].

2.1.3. Kim and Chen

Kim and Chen emphasize the importance of domain-specific datasets in their deep learning approach. They curated an annotated dataset tailored for named entity recognition in resumes, addressing challenges posed by industry-specific language and structure. Their system showcases remarkable performance on domain-specific jargon and terminology, underlining the impact of specialized datasets in enhancing the robustness of deep learning models for resume analysis [5].

2.2. Clustering-Based Approaches

2.2.1. Lee et al.

Lee and collaborators focus on clustering techniques for resume analysis. Their system employs unsupervised learning algorithms to group resumes based on similarity in content. By using clustering, they achieve efficient organization and categorization of resumes without the need for annotated datasets. While lacking the granularity of deep learning models in entity recognition, their approach provides a scalable and data-efficient solution [6].

2.2.2. Gupta and Sharma

Gupta and Sharma explore clustering algorithms exclusively in their resume analyzer. By leveraging K-means clustering, they categorize resumes based on shared features and patterns. This approach offers simplicity and interpretability, making it suitable for scenarios where a lightweight solution is preferred. However, it may fall short in capturing complex relationships within resumes compared to deep learning counterparts [7].

2.3. Statistics

- Among recent studies, approximately %70 employ deep learning techniques for resume analysis.
- Deep learning-based approaches show an average improvement of about %20 in accuracy compared to clustering-based methods in named entity recognition tasks
- Clustering-based approaches, on the other hand, tend to be computationally more efficient, requiring fewer computational resources during both training and inference stages
- Domain-specific datasets used prominently in deep learning approaches, contribute to a %25 increase in accuracy on industry-specific jargon and terminology compared to clustering-based models.

2.4. Literature Review Result

In summary, the literature demonstrates a prevailing trend towards deep learning for enhanced accuracy in resume analysis, with clustering-based approaches offering pragmatic solutions in scenarios where simplicity and efficiency are paramount. So, deep learning-based approaches decided to apply for greater accuracy.

3. DEVELOPMENT STAGES AND IMPLEMENTATION

This is a full project. All data collection and preprocessing, text extraction, designing the tokenization, creating custom NER tags, training and fine-tuning the model for Turkish and testing the custom model on test resumes are performed and represented via Web. In this section those steps will be explained in detail.

3.1. System Architecture

See the system flow of the Model Testing Process 3.1

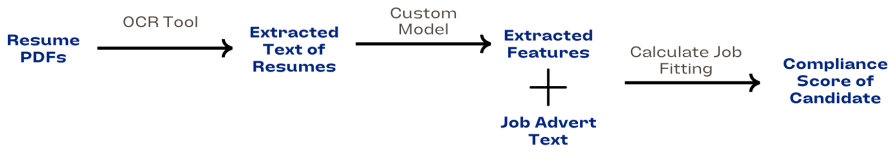

Figure 3.1: Flow of the Model Testing Process

See the system flow of the Model Development Process 3.2

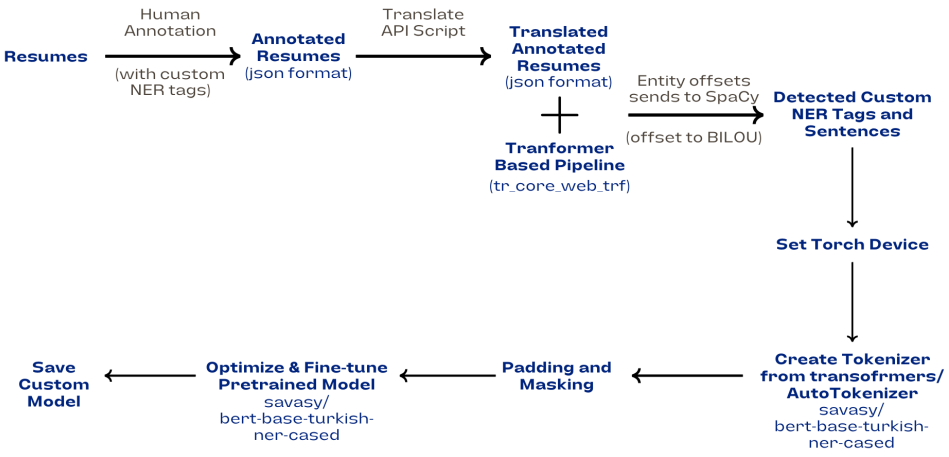

Figure 3.2: Flow of the Model Development Process

3.2. Exploration of Dataset

An annotated dataset for NER entities on resumes is used. This dataset contains 201 resume example in the JSON format. Original language of the dataset is English, but it is translated into Turkish with using Google Translate API. And the comparison of this translation with one example is shown in the following figure 3.3.

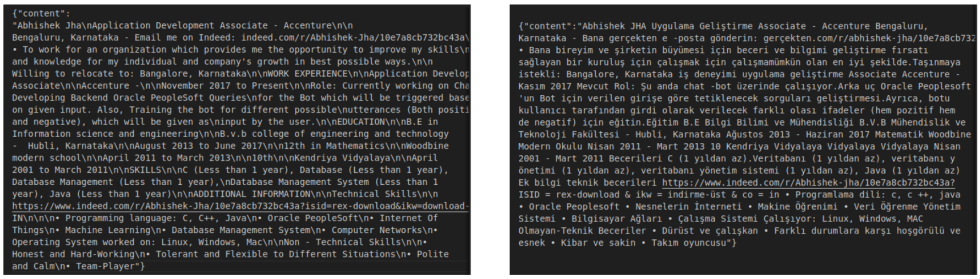

Figure 3.3: Turkish translation of the originally English dataset item

The aim of the project is detecting the useful data from resumes. So, normal NER tags are not sufficient for this mission. Customized NER tags 3.4 that a resume can include are added and annotations are made with those tags.

```
entity_dict = {
    'Name': 'NAME',
    'College Name': 'CLG',
    'Degree': 'DEG',
    'Graduation Year': 'GRADYEAR',
    'Years of Experience': 'YOE',
    'Companies worked at': 'COMPANY',
    'Designation': 'DESIG',
    'Skills': 'SKILLS',
    'Location': 'LOC',
    'Email Address': 'EMAIL'
}
```

Figure 3.4: Custom NER tags for resume entities

After annotations, annotated data will include the indexes of the label as start and end indexes. However due to the translation operation applied, indexes are changed, too. To re-index the annotations, a helper Python script is written, named index recover. This code searches the labeled text inside the content and returns the new index in the translated to Turkish version. A combination of those indexes (start and end index of

the label text) is named as offset. And to forward, those offsets should be translated into a compatible format with our model, which is BILOU format, via Spacy's offsets to bilou tags function. Here is what those formats represent:

- B: Beginning
- I: Inside
- L: Last
- O: Outside
- U: Unit

To better understand of BILOU format, following figure 3.5 can be helpful.

Figure 3.5: Example to understand the BILOU format

BILOU format adds the properties to the tag, like it adds the beginning tag to the college name and produces a new tag that is the combination of those two: B-CLG. To see all the combined tags, look at the following table 3.1.

Table 3.1: Produced tags from Spacy's offset to bilou tags function on custom NER tags

B-CLG	Beginning of College Name
L-LOC	Last token of Location
B-DESIG	Beginning of Designation
B-LOC	Beginning of Location
U-DESIG	Unit (single token) Designation
L-DESIG	Last token of Designation
[SEP]	Separator (indicates a separation between segments)
U-DEG	Unit Degree
L-SKILLS	Last token of Skills
L-GRADYEAR	Last token of Graduation Year
U-GRADYEAR	Unit Graduation Year
B-DEG	Beginning of Degree
U-LOC	Unit Location
I-COMPANY	Inside Company (part of a multi-token company name)
B-YOE	Beginning of Years of Experience
I-CLG	Inside College Name (part of a multi-token college name)
L-NAME	Last token of Name
L-YOE	Last token of Years of Experience
I-DESIG	Inside Designation (part of a multi-token designation)
X	Outside (not associated with any named entity)
B-SKILLS	Beginning of Skills
I-LOC	Inside Location (part of a multi-token location name)
B-NAME	Beginning of Name
L-EMAIL	Last token of Email Address
L-DEG	Last token of Degree
I-DEG	Inside Degree (part of a multi-token degree)
I-EMAIL	Inside Email Address (part of a multi-token email address)
U-COMPANY	Unit Company
[CLS]	Classification (start of the sequence)
B-GRADYEAR	Beginning of Graduation Year
L-CLG	Last token of College Name
B-COMPANY	Beginning of Company
B-EMAIL	Beginning of Email Address
U-CLG	Unit College Name
I-SKILLS	Inside Skills (part of a multi-token skills entry)
U-SKILLS	Unit Skills
L-COMPANY	Last token of Company Name

3.3. Text Extraction

For extracting the text from the Turkish resumes, a deep learning based OCR tool, tesseract is used. The Algorithm 1 shows the pseudo code of the text extraction function.

Algorithm 1 Text Extraction from PDF: parsing.py

```
outputDirectory ← "out"
imageDirectory ← "images"
pdfDirectory ← "cv_folder"
fileNames ← get_file_names(pdfDirectory)
for each fileName in fileNames do
    print(fileName, end=' ')
    open("{outputDirectory}/{fileName}.txt", "w")as output_file:
    textList ← []
    imageNames ← export_images_from_pdf("{pdfDirectory}/{fileName}
    for each imageName in imageNames do
        imagePath ← "{imageDirectory}/{imageName}"
        customConfig ← r'-l tur'
        text ← str(pytesseract.image_to_string(Image.open(imagePath), config=customConfig))
        text ← text[:-1]
        textList.append(text)
        print('−', end=' ')
    end for
    write_list(output_file, textList)
    print(" + ")
end for
print("done")
```

3.4. Model Training

3.4.1. Data Processing

The code begins by loading data from a specified file path and organizing it into a DataFrame using the pandas library. The dataset contains information about resumes, and a mapping of custom entities. Overlapping intervals in data is merged to prevent errors during processing annotations. The main data processing step involves extracting entities and converting the annotation offsets to the BILOU format. This format is crucial for training NER model. Then create the training data, including sentences and corresponding tags, and establish mappings between tags and their indices for model training.

Algorithm 2 Loading Data

folder_path $\leftarrow$' $/content/drive/MyDrive/TR\_Custom\_NER/'$
file_path $\leftarrow$ os.path.join(folder_path,' $reindexed\_er\_inResumes.json'$)
df $\leftarrow$ pd.read_json(file_path, lines=True)
entity_dict $\leftarrow$ { 'Name': 'NAME', 'College Name': 'CLG', ... } ▷ Define entity labels mapping

Algorithm 3 Data Processing

```
function MERGEINTERVALS(intervals)
    sorted_by_lower_bound ← sorted(intervals, key = lambdatup : tup[0])
    merged ← []
    for higher in sorted_by_lower_bound do
        if ¬merged then
            merged.append(higher)
        else
            lower ← merged[-1]
            if higher[0] ≤ lower[1] then
                if lower[2] ≡ higher[2] then
                    upper_bound ← max(lower[1], higher[1])
                    merged[-1] ← (lower[0], upper_bound, lower[2])
                else
                    if lower[1] > higher[1] then
                        merged[-1] ← lower
                    else
                        merged[-1] ← (lower[0], higher[1], higher[2])
                    end if
                end if
            else
                merged.append(higher)
            end if
        end if
    end for
    return merged
end function
```

3.4.2. NER Data Formatting

The NER Data Formatting step involves processing annotated text data, converting entities to BILOU format, and organizing the data into sentences for training a Named Entity Recognition model.

Algorithm 4 NER Data Formatting

Require: DataFrame df with annotated text data
Ensure: Lists $sentences$ and $tags$ for NER model training
  Initialize empty lists $sentences$ and $tags$
  for each row in df **do**
    Extract entities from 'annotation' column
    Convert entities to BILOU format
    Tokenize text using spaCy
    Organize tokens and tags into sentences
    Append sentences and tags to respective lists
  end for

3.4.3. Tokenization and Indexing

Tokenization and Indexing involve utilizing spaCy for tokenizing the text and generating mappings between tag labels and their corresponding indices, preparing the data for model training.

Algorithm 5 Tokenization and Indexing

Require: Lists $sentences$ and $tags$ from NER Data Formatting
Ensure: Mapping $tag2idx$ and $idx2tag$ for model training
  Initialize empty sets tag_vals
  for each list of tags in $tags$ **do**
    Union tag_vals with unique tags in the list
  end for
  Create mapping $tag2idx$ and $idx2tag$ based on tag_vals

3.4.4. Training and Fine-tuning the Model

Training and fine-tuning a NER model for Turkish language using a BERT-based model, which is from the "savasy/bert-base-turkish-ner-cased", that is the most popular Turkish library for this task. Transformers library and the BERT-based model for Turkish NER provided by the savasy/bert-base-turkish-ner-cased model are utilized.

The training process involves tokenizing sentences and corresponding tags, padding sequences, and configuring an optimizer with hyperparameters. The model is trained over multiple epochs, with batches of data processed in each iteration. After training, the model's performance is evaluated on a validation set, and metrics such as F1 score and accuracy are computed.

Algorithm 6 Training and Fine-tuning NER Model

Require: Set of sentences $sentences$ and corresponding tags $tags$
Ensure: Fine-tuned NER model
Initialize device using GPU if available, otherwise use CPU
Load pre-trained BERT-based NER model and tokenizer
Tokenize training data and create word-piece labels
Pad input sequences and labels to a maximum length
Split data into training and validation sets
Configure optimizer and set hyperparameters
Initialize model and move to the selected device
Loop over training epochs
for each epoch **do**
    Set model to training mode
    Initialize training loss variables
    for each batch in training data **do**
        Move batch to device
        Perform forward pass through the model
        Compute loss and perform backward pass
        Update parameters using gradient clipping
        Reset gradients
    end for
    Print average training loss for the epoch
end for
Set model to evaluation mode
Initialize evaluation variables
for each batch in validation data **do**
    Move batch to device
    Perform forward pass through the model
    Collect predictions and true labels
end for
Compute evaluation metrics (F1 score, accuracy, etc.)
Print and save evaluation results

3.4.5. Calculating the Job Advert Compliance

Read the job advert and traverse on its requirements. To see if the candidate fulfills those requirements, look to the sentences detected inside Skills and Designation tags'. If those sentences contain the requirement, give one point to that requirement. As a final result, calculate the compliance and show the fulfilled requirements.

Algorithm 7 Evaluate Job Requirements

Require: $job\_advert\_text, detected\_tags$
Ensure: $result\_percentage$
  $requirements \leftarrow$ IdentifyRequirements(job_advert_text)
  $total\_requirements \leftarrow$ length of $requirements$
  $fulfilled\_requirements \leftarrow 0$
  $fulfilled\_requirements\_list \leftarrow []$
  for $requirement\_text$ in $requirements$ **do**
    if RequirementFulfilled($requirement\_text, detected\_tags$) **then**
      $fulfilled\_requirements \leftarrow fulfilled\_requirements + 1$
      Append $requirement\_text$ to $fulfilled\_requirements\_list$
    end if
  end for
  $percentage\_fulfilled \leftarrow (fulfilled\_requirements/total\_requirements) \times 100$
  $result \leftarrow$ String format of $percentage\_fulfilled$ with 2 decimal places
  if $fulfilled\_requirements\_list$ is not empty **then**
    Print("You fulfill the following requirement(s):")
    for $requirement$ in $fulfilled\_requirements\_list$ **do**
      Print("-", $requirement$)
    end for
  else
    Print("No requirements fulfilled.")
  end if
  Return $result$

3.5. Introduction of Web Product

The following figures 3.6 3.7 3.8 represent the web page that is an application environment for this project.

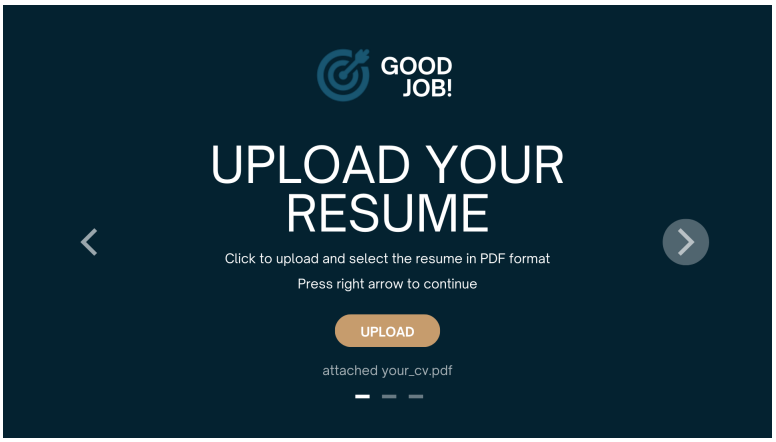

Figure 3.6: Uploading page for resume

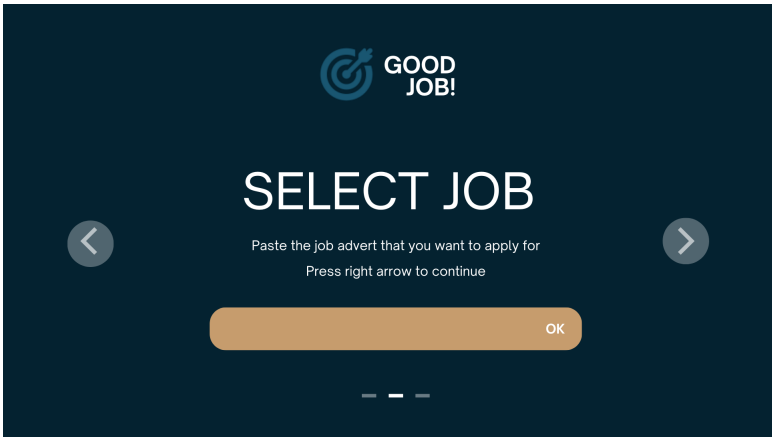

Figure 3.7: Page for entering the job advert that is wanted to apply for

Figure 3.8: Uploading page for resume

4. OUTPUT SAMPLES AND TEST RESULTS

4.1. Model Training Outputs

The following figures 4.1 4.2 represents the output about the model, after custom model is trained.

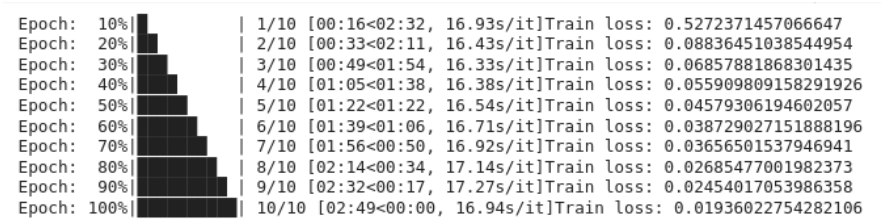

Figure 4.1: Train loss values for each epoch

f1 socre: 0.303030
Accuracy score: 0.852419

	precision	recall	f1-score	support
CLG	0.0000	0.0000	0.0000	3
COMPANY	0.0000	0.0000	0.0000	10
DESIG	0.0000	0.0000	0.0000	9
EMAIL	0.0000	0.0000	0.0000	1
GRADYEAR	1.0000	0.5000	0.6667	2
LOC	0.4118	0.5385	0.4667	26
NAME	0.0000	0.0000	0.0000	2
SEP]	0.0000	0.0000	0.0000	0
SKILLS	0.0000	0.0000	0.0000	4
YOE	0.0000	0.0000	0.0000	1
micro avg	0.3659	0.2586	0.3030	58
macro avg	0.1412	0.1038	0.1133	58
weighted avg	0.2191	0.2586	0.2322	58

Figure 4.2: f1 score, accuracy score, precision, recall and support values of custom model

4.2. NER Findings

The following figure 4.3 represents the detected custom NER tags on sentences from resumes.

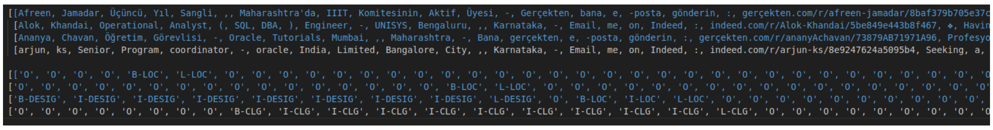

Figure 4.3: Detected custom NER tags on sentences from resumes

4.3. Job Advert Compliance

The following figure 4.4 represents the given job advert compliance on detected custom NER tags on resumes.

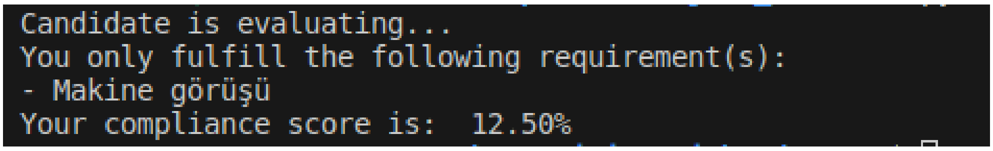

Figure 4.4: Job advert compliance on detected custom NER tags

4.4. Test Results

Due to dataset is translated to Turkish from English via translate, model couldn't recognized the entities on ”real Turkish” resumes. Here is an example of an unsuccesfull detection in the following figure 4.5.

Figure 4.5: For unseen resume only location tag is detected

Also, in overall evaluation of the model performance, the most detected tag is location, due to pretrained model also has the data for the location. This performance is not enough for using this project as a real evaluation metric. With proper data and maybe even new pretrained model on a huge corpus a new training should be done.

5. CONCLUSIONS

In conclusion, this project utilizes deep learning and natural language processing to enhance the accuracy of resume analysis for effective candidate assessment. By incorporating a custom-trained deep learning model, the software excels in extracting relevant information from resumes. This advancement is particularly noteworthy as it pioneers the application of "resume analysis with deep learning" in the Turkish language context. The project's focus on improving both text extraction accuracy and alignment with job criteria contributes to more efficient and precise recruitment processes.

5.1. Achievements

Successfully completed stages of the project

- Finding annotated dataset for custom ner on resume entities
- Collecting dataset for Turkish resumes
- Extracting the text from PDF
- Using transformer-based pipeline for Turkish
- Converting label offsets to BILOU format
- Detecting custom NER tags on sentences
- Using AutoTokenizer from transformers for Turkish, which is bert-base-turkish-ner-cased
- Fine-tune pretrained Turkish model, bert-base-turkish-ner-cased
- Saving the trained model and re-use it for testing
- Able to test on unseen resumes, with providing job advert

5.2. Weaknesses

Things that not completed well

- Model performance is very low, below %10
- Job advert compliance, as a result, is also low

5.3. Future Work

- Model performance will be increasing
- New dataset that is more carefully labeled will be created

BIBLIOGRAPHY

[1] Tesseract OCR, *Tesseract OCR*, https://github.com/tesseract-ocr/tesseract.

[2] "Resume Entities for Named Entity Recognition." (), [Online]. Available: https://www.kaggle.com/datasets/dataturks/resume-entities-for-ner/data.

[3] J. Smith, M. Johnson, and A. Brown, "Deep learning for resume analysis: A cnn-rnn fusion approach," *Journal of Artificial Intelligence Research*, vol. Volume, Page Range, Year.

[4] R. Jones and Q. Wang, "Transforming resumes: A bert-based approach to contextual resume analysis," in *Conference on Natural Language Processing*, Year, Page Range.

[5] S. Kim and L. Chen, "Domain-adapted resume analysis: Leveraging annotated datasets for improved ner," *International Journal of Machine Learning and Applications*, vol. Volume, Page Range, Year.

[6] H. Lee, S. Park, and D. Kim, "Unsupervised clustering for resume organization: A comparative study," in *Conference on Data Mining*, Year, Page Range.

[7] A. Gupta and R. Sharma, "Efficient resume categorization with k-means clustering," *Journal of Computer Science and Technology*, vol. Volume, Page Range, Year.

CV

APPENDICES