UNIVERSITY OF HERTFORDSHIRE

School of Physics, Engineering and Computer Science

MSc Artificial Intelligence and Robotics

7COM1086-0509-2020: Artificial Intelligence and Robotics Masters Project

27th August, 2021

Automated Resume Evaluation using Machine Learning and Deep Learning Algorithms

Name: Atik Jain

Student Id: 18036573

Supervisor: Lili Kirner

Declaration

This report is submitted in partial fulfillment of the requirement for the degree of Master of Science in Artificial Intelligence and Robotics at the University of Hertfordshire (UH).

It is my work except, where indicated in the report.

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on the university website, provided the source is acknowledged.

Acknowledgement

I want to grab the opportunity to express my gratitude towards my supervisor, Lili Kirner, for her guidance. Her continuous support helped me in the completion of my dissertation.

Moreover, I want to thank and extend my appreciation to my friends. They have always had complete confidence in me. I have been able to get through the complex programming with their help. I have also constantly gone through the discussion to understand the errors and the debugging throughout the software development.

Most importantly, I ought to express heartfelt gratefulness to my parents for their constant encouragement, support, and best wishes for my higher education and MSc project.

List of Abbreviation

- NLP – Natural Language Processing
- AI – Artificial Intelligence
- ML – Machine Learning
- DL – Deep Learning
- NER – Named Entity Recognition
- ARES – Automated Resume Evaluation System
- BERT – Bidirectional Embedding Representative Transformer
- CRF – Conditional Random Fields
- LSTM – Long Short-Term Memory Classifier
- NLTK – Natural Language Toolkit
- CNN – Convolutional Neural Network
- IPR – Interim Project Report
- FPR – Final Project Report
- DPP – Detailed Project Proposal
- MAXENT – Maximum Entropy
- API – Application Programming Interface
- TPU - Tensor Processing Units
- CCG – Combinatory Categorical Grammar
- BiLSTM – Bi-directional LSTM
- MLM – Masked Language Modelling
- NSP – Next Sentence Prediction
- BILOU – Beginning Inside Last Outside Unit
- HMM – Hidden Markov Model
- MEMM – Maximum Entropy Markov Model
- HR – Human Resource

List of Figures

List of Tables

ABSTRACT

The report is based on a project which aims to use Machine Learning models to solve a modern, real-time issue with the current recruitment process around the globe. As the COVID-19 pushed the world towards an inevitable pandemic, every economic sector was hit and had to adjust the modus-operandi to stay relevant. The human resources sector, closely related to direct or indirect employment, was also hit, processes like profile shortlisting, interviews were shifted towards online platforms. Processes like shortlisting profiles from hundreds of resumes could prove to be tedious, increasing the scope of error or in the worst case could even lead to unwanted filtering of the right talent.

Since machines are not prone to mental fatigue while performing the same set of tasks over and again, using them to reduce human efforts at the early stage of filtering resumes could greatly eliminate the scope of error and reducing the number of unwanted filtering of the right talent. The idea proposed in this report is to develop a smart ML-powered system that could effectively understand English to extract information present from a pool of resumes and rank the resumes according to how relevant the resumes are to the Job advertised by a dummy organisation. To develop a system with language interpretation ability, a good amount of knowledge about the Machine Learning models is required. To give a perspective on the existing works, this report includes a detailed section on “literature review” which includes an extensive review of a good deal of research papers. The research aims to perform Name Entity Recognition (NER) on differently structured resumes to extract important information from them. The research aims to find ways NER could be used for extracting keywords (Skills) with good accuracy in efficient way with the given hardware. The research attempts to provide a solution to the employers to choose the best profile to interview during the worldwide crises while raising some research questions.

To demonstrate and evaluate the knowledge acquired during the research phase, the report includes a chapter providing shreds of evidence in the form of code snippets and the results, in the form of accuracy. Once, all the performances have been evaluated, the evidence of implementation of the smart system has been included in this report. The results from the implementation of the code for different ML models, combined with the research discussed in this report seek to answer the research questions.

Table of Contents

Chapter 1 – Introduction

1.1 Background

Back in 2017, technology changed and so did the way of applying for Job. But oddly, while electronic means of collecting resume has made job hunting somewhat easier, in many ways the experience remains the same for the employer and hiring a relatively good candidate for a job became difficult for an organisation owing to the enormous CVs/resumes received, which, are not always structured and comes in all sorts of digital formats like word files, pdfs, otds and many more.

In 2019, the world was oblivious of the covid-19 crisis and the impact of the pandemic could have on the recruitment scenario. Every single country was affected because of the crisis. The restrictions and lockdowns were imposed to control the spread of virus, which severely affected different businesses and reduced the employment rate. In April 2020, the unemployment rate reached to astonishing 14.8% in the US alone—the highest rate observed since data collection began in 1948. (Falk, 2020)

As the world embraced the pandemic, complying with the regional government restrictions, the ways of recruitment were also changed, causing changes in how people apply for jobs. There has been increasing growth in online recruitment, one can see that the traditional hiring methods were rendered inefficient. The existing conventional technique, usually include manual reviewing of the pool of resumes and then producing a shortlist of resumes for eligible candidates to be interviewed. The types of resumes received by a company may fall into one or more category(s) like, the written account resumes, the targeted resumes, the visual resumes, the practical resumes, and more; Structure is another time-consuming issue—when extracting useful information.

Since, the process of shortlisting the resume is tedious and may put an HR team under great mental stress, from the computer science point of view a need for some degree of automation could be sensed. To precisely understand the situation, there is a need to overcome the shortlisting problem for the employer to process a bulk of resumes to choose the perfect candidate with the required skills. Natural Language Processing (NLP) could play a vital role in the task. Text Classification, Text Mining, Data filtering, Stemming, and Lemmatisation are some of the crucial processes that could be achieved using advanced approaches in Machine Learning and Deep Learning. Employing such techniques could effectively address the issue of extracting skills from a pool of resumes, automating the part to select resumes with suitable skills mentioned, in the shortlisting process. Resume Screening and Resume Evaluation are the terms used for identifying the right candidate and measuring the resume whether it fits for the job and the organisation. When analysing the term resume evaluation, HRs circle around the same concept of skills matching with the job description. Extracting featured/important information from the file and classification is what comes under the NLP term Named Entity Recognition.

Named Entity Recognition (NER)—sometimes also referred as entity chunking, entity extraction, or identification—the process of identifying and categorising key information (entities) in texts. The term “Entity” refers to a piece of meaningful and valuable information or keywords that makes a prediction syntactically and semantically correct. To train NER for

extracting valuable information there are multiple machine learning models available, such as Spacy, Conditional Random Fields Classifier (CRF), Long Short-Term Memory Classifier (LSTM), Natural Language Toolkit (NLTK), Bidirectional Encoder Representations from Transformer (BERT) and many more. The scope of study of ML models is limited to Spacy, CRF, and BERT due to a time and resources constraint.

The different stages involved to automate the NER process are listed below:

- Data collection from Kaggle of different Resume. Kaggle is an open-source code and data-sharing platform and is used by machine learning enthusiasts.
- Pre-processing of the dataset for filtration of noise and duplicates.
- Feature Extraction (Important Information) and text classification using BERT, Spacy and CRF model.
- Implementation of Automated Resume Evaluation System (ARES) using BERT.

The research includes the study of the unsupervised data using ML models for entity extraction and classification of the data collected from Kaggle. Kaggle allows users to use and publish datasets, community support for data-sciences and build models on a web-based data-science environment without copyright issues.

1.2 Research Question

The research has proposed an adequate model to extract valuable/featured information from the given set of resumes and rank the resumes according to the job requirements advertised by a hypothetical organisation. The model will help the organisation to shortlist the candidate efficiently, introducing an automated system for resume processing. As the research circulates around NER, the report should address the following questions:

- Which ML model among Spacy, CRF, and BERT, is more efficient and accurate for handling entity extraction and classification given a limited size of data?
- Does Bidirectional Encoding Representation for Transformer (BERT) have any real-time advantage over other models such CRF and Spacy when the epochs are increased, in resume evaluation?
- Does training an ML model multiple times improves the performance of the model?

The focus of this research is to study the accuracy and precision of the BERT model introduced by google in 2019 against the previously existing model like CRF and Spacy; drawing the alternative hypothesis that overall accuracy and precision can be enhanced by using BERT for Resume Evaluation. The complexity of the data and noise could also adversely affect the precision and accuracy parameters of a model and is the null hypothesis for this research. The alternate hypothesis validation could be done by comparing BERT with previously mentioned model.

1.3 Aim

The research intends to find the best machine learning model for resume parsing (entity extraction and classification) by comparing different classifier models such as BERT, Spacy, and Condition Random Fields Classifier (CRF). The aim could be divided into the following:

1. Increase the precision and accuracy of entity extraction and identification.
2. Noise filtration using from the data and perform NER using BERT, Spacy and CRF.
3. Using the trained NER model on the unsupervised data (resumes) and produce an accurate ranked list of resumes.

1.4 Objectives

To prove the alternative hypothesis discussed previously, a set of short-term goals split into two groups i.e., core objective and advanced levels based on their priority and complexity, have been listed in this section.

1.4.1 Core Objectives

1. Research on existing works and distributed papers related to NER.
2. Research on some existing ML models for NER.
3. Shortlisting the ML models to implement based on the most common computer hardware available.
4. Pre-processing the unsupervised data using Text Classification, Text Vectorisation, Stemming, and Lemmatisation techniques.
5. Filtering out the noise/unwanted info from the data and null entries from the data.
6. Precisely choose the test model from an unsupervised data.

1.4.2 Advanced Objectives:

1. To implement the pre-trained BERT model and fine-tune for resume evaluation.
2. Develop the NER model for accurate resume evaluation with BERT, Spacy and CRF.
3. To compare the Resume Evaluation results of the tested models. i.e., BERT with the other models such as Spacy and CRF.

1.5 Project Plan

The following table has total task and their detailed information and the progress that is made according to the timeline. The objectives were needed to be rechecked after studying relevant research papers. The research question remains the same since the research is focused to develop a model for resume evaluation to achieve higher performance.

No	Task Name	Task Details	Allocated Dates
1.	Literature Review	Study the previous work done related to Resume Parser and Named Entity Recognition.	01/05/2021
2.	Study of Relevant topics	Study of topics related and helpful for the research such as BERT, Spacy, CRF, and Named Entity Recognition.	07/05/2021

3.	Preparing Detailed Project Proposal	Description of the project idea on the questionnaire.	15/05/2021
4.	Development of Research Questions	A thorough study of the technologies and work done for the Research Questions.	21/05/2021
5.	Break Down of Aim and Objectives	The structural Breakdown of Aim and Objectives to achieve the research/study's goal.	27/05/2021
6.	Submission of DPP and discussion.	Submission of detailed project proposal and discussion with the supervisor about the idea and research on it.	11/06/2021
7.	Collection of Relevant Data	Searching and Selection of relevant Dataset for the research.	13/06/2021
8.	Pre-processing of Data	Noise filtration on the data and pre-processing. Converting unstructured to Structured format.	22/07/2021
9.	Preparation and Submission of Interim Progress Report (IPR)	Submission of IPR and discussion about the progress till date and future plan of action.	02/07/2021
10.	Implementation of Classification Techniques	Classification of named entity using BERT and documenting the progress.	05/07/2021
11.	Comparison of Different Machine Learning Algorithms.	Implementation of different ML models (Spacy, CRF) on the same dataset to find the best fit model to provide more accuracy and efficient results.	09/07/2021
12.	Testing of Working Model	Integration of whole code and Testing of code modules in the final working model.	12/08/2021
13.	Work on the Final Project Report (FPR)	Documenting all the information about the relevant research and experiments in Final Project Report.	15/08/2021
14.	Submission of FPR	Submission of the working model (code) and the FPR.	27/08/2021

Table 1-1 Planned Project Timeline

1.6 Chapters Overview

The complete research is organised in the chapters as follows:

- Chapter 1 – Introduction
  The introduction chapter lays foundation of the preceding research and development work while introducing the overall aim and short-term objectives. The chapter also has a dedication section to discuss about the legal, social and ethical issues. The research questions are also discussed.

- Chapter 2 – Literature Review
  The section is concerned with the discussion on some of the existing works on resume parsing, in broader term the Named Entity Recognition. This chapter also provides the background information of the major technologies used throughout the research.

- Chapter 3 – Research Methodology
  The chapter is concerned with a discussion on the tools and resources used for the purpose of the research. The chapter has a generous amount of information about various NER techniques and methodologies, providing an essence of how the underlying technologies could be implemented together. The chapter concludes by reviewing the steps/plan for the research.

- Chapter 4 – Experiments and Results
  The chapter provides a walk-through of the steps followed to implement CRF, BERT and Spacy model while employing the information gained from the previous sections. The chapter is one of the focus of the research as the results obtained in this chapter provides the necessary information required for drawing the conclusion of the project while extending the scope for discussion on what could be improved in the future works.

- Chapter 5 – Conclusion and Future Work
  The chapter is the concluding chapter of this report and tries to answer the research question introduced in the previous chapters. The hypothesis and the null hypothesis are revisited to check whether enough evidence was gathered. The chapter includes what could be improved in the resultant product based on the results.

1.7 Consideration of Ethical, Legal, and Social Issues

1 The Data Protection Act 2018 is the UK's equivalent of the General Data Protection Regulation (GDPR) (Regulation, 2018). Under DPA 2018, anyone using any personal information must abide by the rules called "*Data Protection Principles"*. Data that could lead to the identification of an individual, or a group of individuals, must be avoided.

2 The fairness of the system shall not be affected by the following:
  2.1 Race,
  2.2 Nationality,
  2.3 Caste,
  2.4 Sex,
  2.5 Ethnicity

  of the candidates. The resume processing shall be done to show the transparency of our project and the work performed by it as mentioned in the Data Protection Act 2018.

3 All the subcodes and supporting third-party libraries shall be used only and only when the license is open-source free to use and free to distribute, preventing copyright issues.

Chapter 2 – Literature Review

2.1 Motivation

Automated Resume evaluation or resume parsing refers to the automated storage, organisation, and analysis of job resumes. Companies or organisations can use resume parsing software to quickly discover keywords, skills, and other important information to sort through enormous number of applications with different types of formats and find the most suitable candidate for the job. Extracting the featured information from the whole document (resume) can be achieved using Natural Language Processing. Though the concept appears to be simple, there are several features for which researchers have been constructing increasingly realistic models. Due to the adverse effect of the pandemic, the world has seen sudden changes in almost every economic sector. This affected the way of people looking and applying for jobs. Since everyone is looking for jobs online, the pressure on the recruitment team has come to an extreme limit because of the bulk of resume they receives for the same job role.

The motivation behind the research is the state-of-the-art results of the Bidirectional Embedding Representation Transformer (BERT) achieved by multiple data scientist and researcher. Google first introduced the Transformers in 2017. At the time of their introduction language models used Recurrent Neural Network or the RNN and Convolutional neural networks or the CNN, to handle tasks in NLP domain. Resume evaluation using BERT is one of the fields in NLP that requires more research and extensive knowledge of transformers.

2.2 General Terminologies in Background Research

2.2.1 Morphological Analysis

The systematic study of how misspelling of words could lead to different meanings in a sentence is called Morphology in NLP. The study follows the concept of Spelling check by the means of popular methods like Insertion (mistyping ‘the’ as ‘th’), Substitution (‘the’ as ‘thw’), and Deletion (mistyping ‘the’ as ‘ther’).

2.2.2 Syntactic Analysis

Syntax of a human language is a related set of rules which defines how words should be arranged to give meaning to a sentence. The syntax is different for different languages, the main goal of syntactic analysis is to look whether the sentence follows the grammatical rules of the language of interest. It is also used to recognise the text and assign token or small word groups into grammatical phrases, and to allocate it with a syntactic structure.

2.2.3 Semantic Analysis

Semantic Analysis is the study of meaningful and non-meaningful sentences. Semantic analysis is important in the sense that a syntactically correct sentence may not bear any meaning. Thus, for a computer software should be able to understand the syntax and semantics of a language.

2.2.4 Tokenisation

Tokenisation is the process of selecting a sequence of strings into keywords, symbols, or words; the selected sequence is called a token. The token is removed from a test sentence fed to an ML model, to check whether the model can “guess” the token further verifying if the model has acquired correct knowledge of syntax and semantics of the test language.

2.2.5 Parsing

The process of converting raw data/information into a data that could be analysed and manipulated by a piece of code is called data Parsing.

2.2.6 TPUs and POD Configuration

“Tensor Processing Units (TPUs) are Google’s custom-developed application-specific integrated circuits (ASICs) used to accelerate machine learning workloads. TPUs are designed from the ground up with the benefit of Google’s deep experience and leadership in machine learning.” (Cloud, n.d.)

PODs are used in cases where multiple independent containers are required to execute, and the result is a collective outcome of processed containers. The POD also features an ability to run various independent pipelined routines of a single program where program specific containers are tightly coupled and work on common resources.

2.3 Background Research

(Nimbekar, 2019) and her team proposed an idea of an Automated Resume Evaluation system in an IEEE research paper. The whole process in their research was divided into 3 phases. The three phases included, the conversion phase, the purpose was to convert all the unstructured data to a structured data for the model to process. This phase removed all the unwanted new lines, spaces, and special characters. The second phase also called the Extraction phase served the process of fetching all the information from the resume and label them using Named Entity Recognition (NER), a pre-defined dictionary. The extracted information was passed to the

last phase also called the Filtration Stage. The Filtration stage processes the data and removes all the non-relevant information from the dictionary received from the previous stage. Collaborative filtering was used to provide with all the applicants who matched the job description. The filtration process was made more efficient, a score was given to each resume according to the skills matched to rank the applicant. Section Based Segmentation module was used to parse the whole resume. The segmentation is processed sentence by sentence. The whole comparison was done using semantic network and ML algorithm ‘Learning to Rank’ was used to rank the resume.

(Zubeda, 2016) in her project report Resume Ranking using NLP and ML has followed three analysis methods, i.e., Morphological Analysis, Syntactic Analysis, and Semantic Analysis to do the parsing of the whole document. Natural Language Toolkit (NLTK) was used to extract the entities from the document in the project. The model proposed by them accepted data in all types of file format and converted the data to an HTML based data structure, accessible via an https link, to be accessed using Scrapy web-crawler. NLTK was used to serve four purposes, i.e., Simplicity, Consistency, Extensibility, and Modularity. The main

drawback of the project, the data was required in web-post link format for the model to parse it.
The term which came to light from the study was Natural Language Toolkit (NLTK), which could be used as one of the ML tools to compare while implementing our proposed model. NLTK is a set of tools used for building a python program that works with human language data for applying in statistical natural language processing (NLP). NLTK also provides different in-built libraries for tokenisation, parsing, classification, stemming and tagging. The drawback of NLTK is, that NLTK is meant for research and teaching purposes. So, to perform different NLP tasks, a good knowledge of NLTK libraries is required.

A model was also proposed by (Dragoni, 2016) to look for syntax and semantic relations as the relation between words defines a better explanation of why they were used. According to (Dragoni, 2016), a single NLP model approach is not satisfactory for better results. (Dragoni, 2016) combined different models to implement for rule extraction from a set of Legal Documents. The semantics and syntax (spelling and grammar check) were performed by using Stanford parser. The concept of Combinatory Categorical Grammar (CCG) was enforced to extract logical dependencies in the legal documents. While standalone OS-based programs were being studied in abundance a web-based application was also developed for Job seekers using Spacy and NER by (Amin, 2019). The Job portal was based on extracting valuable information from LinkedIn using an API and checking it with all the Jobs available on the market. NER was used to extract information related to the Experience required, education, and skills, and Spacy was used to convert the file into text format. The model had an accuracy of 67% because the dataset used for training was not pre-processed and contained noisy data in abundance. The project was not a success as it was rejecting the applications if the format was not a perfect match to the format used for training the model. The project highlights the core issues of input data mismatch in ML models and the importance of cleaning noisy data before passing them for training a model. The study also shed light on the different in-built libraries of Spacy and how Spacy libraries could be used to process different formats of data and perform data visualisation.

In 2019, Google first announced a rather radical idea of multiple fed systems, consisting of an array of encoders that could process multiple streams of data at once from both directions (into the encoders and out from the encoders). Such systems at early stages showed great potential specifically in NER, (Han, 2020) added their research and knowledge about BERT and how BERT could be used at the document level in their paper, *“A novel document-level relation extraction method based on BERT and entity information”*. A document-level entity mask method was used to mask each mention of the entities by special tasks. Sentence level relation extraction was done to fetch relative information from a whole document. They have also compared different ML models with the result of BERT to find the best entity extraction algorithm. The compared models were the Convolutional Neural Network (CNN) model with an accuracy of 42.26%, LSTM with an accuracy of 50.07%, Bi-directional LSTM with an accuracy of 51.06%, and BERT model with an accuracy of 54.83%. The research by (Han, 2020). They provided an in-depth knowledge of Masking and how it helps for pre-training of the BERT model. From the research, it was evident that BERT outperformed CNN, LSTM, and Bi-directional LSTM with a good margin.

(Devlin, 2018) provided a deep understanding of BERT transformer and its type in their research paper “BERT: Pre-training of Deep Bidirectional Transformer for Language

understanding". Their research was based on understanding language for the prediction task. The whole process was divided into two tasks. First was Masked Language Modelling (MLM), in this task, 15% of the words in each sequence were replaced with a Mask token and the model was trained with this sequence to predict the missing words based on the sentences surrounding the masked entity. The second task was the Next Sentence Prediction (NSP), the purpose of NSP was to check if the two statements which are placed together have any relation among them or not. (Devlin, 2018) used pre-trained BERT model for the research and had performed the implementation for both algorithms i.e. BERT small and BERT large. BERT large has 24 layers, 1024 hidden, 16-heads, 336M parameters which resulted in better accuracy of 82.1% as compared to BERT small (12 layers, 768-hidden, 12- heads, 110M parameters) (Pretrained models, 2019) with an accuracy of 79.1%. They also defined the data batch size to 16 and 32. The whole implementation was performed on 4 cloud TPUs in POD configuration.
The research had developed a keen interest to learn about types of BERT tasks i.e., MLM and NSP, and how they can be implemented for the proposed idea.

A better explanation of BERT and how it could be used for a particular task was given by (Bhatia, 2019) in their research paper "End-to-End resume parsing and finding candidates for a job for a description using BERT". They implemented a resume parser using BERT. The whole process was divided into two tasks, the resume parser was used to extract all the information from the document, and it was labelled using NER which was trained using BERT. They have also fine-tuned the pre-trained BERT model with just an additional output layer to create a state-of-the-art model to perform entity extraction from resume. The drawback for the proposed model was, the model was only implemented on LinkedIn resume datasets which are of the same format, so it was unclear if the trained model will show the accurate result if the dataset would change with any other unstructured format.

(Sonar, 2012) also proposed the idea of resume parsing using Named Entity Clustering in their research paper "Resume Parsing with Named Entity Clustering Algorithm". They divided the whole process into 4 stages i.e., Text Segmentation, Named Entity Recognition, Named Entity Clustering, and Text Normalisation. In their research, the text segmentation phase worked on the fact that each heading in the resume contains a block of related information following it. Because of this, the data could be classified under different information types, such as education, work experience, contact information, and many more. A data dictionary was used to store common heading from a resume which are more likely to occur in the resume. During extraction, every heading was considered as a segment and all the information between two consecutive segments are placed under the first heading or the segment. For each segment, a group of Named Entity recognisers were created which were called as Chunker, that works only for that segment. The data was then passed to Named Entity recognition stage, where all the information is extracted and labelled according to the predefined NER library. The output of NER was the entity name, its type, and its start and end position.

The data was then sent to the next stage, which is the Named Entity Clustering stage. In this stage, all the Extracted Entities are grouped in different clusters based on related information. Preceding further, the data was then sent to the last stage also called the Text Normalisation Stage. In the Text Normalisation Stage, all the Noisy data gets filtered from the file. The performance evaluation was calculated using three formulae:

“Precision measures the number of relevant items retrieved as a percentage of the total number of items retrieved.” (Teufel, 2018)

$$Precision = \frac{\#(relevant\ items\ retrieved)}{\#(retrieved\ items)} = P(relevant|retrieved)$$

“Recall is defined as the number of relevant items retrieved as a percentage of the number of relevant items in the collection”. (Teufel, 2018)

$$Recall = \frac{\#(relevant\ items\ retrived)}{\#(relevant\ items)} = P(retrived|relevant)$$

The harmonic mean of precision and recall is taken as F-measure.

$$Fmeasure = \ 2 * \frac{(Precision * Recall)}{precision + recall}$$

The F-measure of the model was between 95% to 100%, which was better than the data ever recorded previously in the same research paper. The disadvantage of the model was that it was unable to process any data which is not in the format of docx or text. The result also differs when worked in a real environment as the resumes received for job descriptions do not follow the same format and do not have the same heading. This resulted in the failure of the proposed model by (Sonar, 2012) as the auto-detect feature might show errors, and many important features were getting rejected because of their uniqueness.

The data collected from the above research makes it clear that a pre-declared or already created dictionary cannot be used to train ML models for real-time entity extraction and classification. Also, the study cleared the fog from the concept of Precision, Recall, and F-measure and how they could be calculated for all the models to compare the best among them for the proposed idea implementation.

(Nongmeikapam, 2011) explained the Conditional Random Fields (CRF) classifier based Named Entity Recognition in their IEEE research paper “CRF based Named Entity Recognition (NER) in Manipuri: A highly agglutinative Indian Language”. The concept of CRF was developed to calculate the conditional probabilities of values on other designated input nodes of undirected graphical models. Their team has performed CRF to understand Manipuri Language and got results as Recall=81.12%, Precision=85.67%, and F-score=83.33%. While (Nongmeikapam, 2011) explained the working of CRF, and the result acquired for NER using only CRF. A combination of neural networks and different machine learning for efficient Resume parsing was also performed by (Ayishathahira, 2018) in their paper ‘Combination of Neural Network and CRF for efficient resume parsing’, (Ayishathahira, 2018) mentioned that the CNN model was used for clustering different segment in a resume. They also used a pre-trained glove model for word embedding. Two different ML models were combined with CNN for better performance. The two ML models were CRF and LSTM. Both models were used for sequence labelling to tag different entities. The outcome of their experiment implementation was CRF+CNN with an accuracy of 78% and Bi-LSTM+CNN was 66%. This enlightens us about the concept that combining two

different approaches does not ensure results with better accuracy. These two research papers “CRF based Named Entity Recognition (NER) in Manipuri: A highly agglutinative Indian Language” and “Combination of Neural Network and CRF for efficient resume parsing” introduced two new Machine Learning models i.e., CRF and LSTM, that could be used in the implementation of the proposed idea and compare the results of with other ML models outcome to find the model with the best performance when worked on the real environment with different structured resumes.

Chapter 3 – Research Methodology

The Automated Resume Evaluation System (ARES) can also be called resume parser and interpreter, this is an extraction and an identification task in NLP requiring a sequence of interdependent processes to be performed on the dataset. Thus, ARES core feature is an extraction and identification task performed on the dataset in JSON format from Kaggle. This chapter briefly describes the methods used to pre-process the dataset according to the different Machine Learning models such as BERT, Spacy, and CRF. This chapter further explains the methodology using the advanced technologies based on what the research has been constructed and designed for Automated Resume Evaluation System, such as different ML algorithms like BERT, Spacy, and CRF. The tools and technologies used have been explained in detail with the model architecture. This section proposes methodologies to identify whether the implementation of ARES using BERT has any real-time advantage over other models such as CRF and Spacy when the epochs are increased.

3.1 Data Collection

The proposed research has been conducted on two different datasets i.e., one dataset "Resume Entities for NER" contains information of multiple resumes and information of all the ann_labels in it with their start position, end position, and the text related to it as shown in Fig.3.1. The dataset is published by DataTurks as an open-source dataset and contains information of 220 resumes. It needs pre-processing before it could be used to train different ML models as it contains a lot of stop words, null values, overlapping entities, and escape sequences (4.2 Data Cleaning and Pre-processing). This dataset will be used for the training and testing of machine learning models like BERT, CRF, and Spacy to use them in a real environment for entity extraction and classification on multiple unstructured resumes.

```
['Afreen Jamadar Active member of IIIT Committee in Third year  Sangli, Maharashtra - Email me on Indeed: indeed.com/r/Afreen-Jamadar/8baf379b705e37c6
 {'entities': [[1155, 1199, 'Email Address'],
   [743, 1141, 'Skills'],
   [729, 733, 'Graduation Year'],
   [675, 702, 'College Name'],
   [631, 673, 'Degree'],
   [625, 629, 'Graduation Year'],
   [614, 623, 'College Name'],
   [606, 612, 'Degree'],
   [104, 148, 'Email Address'],
   [62, 68, 'Location'],
   [0, 14, 'Name']]}]
```

Figure 3-1 Structure of Training Dataset

The second dataset "Resume Dataset" contains multiple resumes to check the implementation of the trained models in a real environment and predict the skills from the resumes. This dataset is provided by Palak Sood on Kaggle, and it contains 238 resumes with different structures. All the resumes are provided in a text format. The entries in the underlying dataset will be passed directly to the trained ML model as it should be able to handle unstructured files and perform resume evaluation and ranking automatically.

3.2 Tools and Technologies

- The research uses Google Collaboration Environment as virtual computer with Jupyter-style IDE, as the BERT model implementation requires a high-powered and high memory GPU, which are available at Google Collaboration for developers.
- Python version 3.6.5 would be the core programming language to implement the models.
- The most common toolkits and libraries used to perform NLP tasks are listed below:
    - NLTK – NLTK is a leading library for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenisation, stemming, tagging, parsing, and semantic reasoning. (Teufel, 2018)
    - NumPy – NumPy is a python library that provides a multidimensional array object, various derived objects (such as masked arrays), and an assortment of routines for fast operations on arrays. It also provides an alternative to the regular python list (NumPy treats an n-embedded lists structure as an n-Matrix) and can also analyse the data much more efficiently than using normal lists. NumPy also performs calculations over entire arrays.
    - Pandas – Pandas is a python library which has an in-built set of tools used to perform data analysis. Most of the model implemented to process data in python has to use Pandas. Pandas can be used to load, prepare, manipulate, model, and analyse data.
    - Scikit-learn – A data mining tool with robust APIs for data analysis. Scikit-learn is built on NumPy, SciPy, and Matplotlib.
    - Seqeval: “Seqeval is a python framework for sequence labelling evaluation. It can evaluate the performance of chunking tasks such as named-entity recognition, part of speech tagging, semantic role labelling, and so on.” (Ramshaw, 1995)
    - PdfMiner: The python package provides a way to extract texts from PDFs which may or may not have been originated from an image.
    - TQDM: It is a library in Python which is used for creating Progress Meters or bars. Implementing TQDM can be done effortlessly with Pandas.
    - Transformers: The idea of transformer is usually used in electronics where multiple encoders and decoders are combined to extract information. The “transformers” library available for python comprises of coded encoder functions connected in a way to mimic the functionality of the pre-trained BERT model which, could be used to perform tasks on texts such as classification, information extraction, question answering, summarisation, translation, and more in over 100 languages.

3.3 BILOU

The library BILOU stands for Beginning Inside Last Outside Unit. BILOU tags are added to find the location of different entities in the document and if they are related to the sentence around them.

If a single entity is recognised while reading the whole line, the tag U (Unit) is added to it.

For a statement with more than one word. The starting entities are tagged as B (Beginning) and the ending entities are tag as L (Last). All the entities between B and L are considered as I (Inside).

Any entity which does not belong to any tags (B, I, L, U) is tagged as O (outside) and such entity is ignored while training the model owing to the fact, they won't change the meaning of a sentence.

A Simple example is considered below to explain it better.

(Works at Gogole) (Gogole) yes yes yes (Gogole Scholar)

Can be encoded with BILOU as

B-Works, I-at, L-Gogole, U-Gogole, O, O, O, B-Gogole, L-Scholar

3.4 Spacy

Spacy is a open-source library, free to use for advanced Natural Language Processing (NLP) with APIs existing for Python. Whenever any text is passed to Spacy, it first tokenises the text to produce a DOC object. The DOC object is then processed in several different sequential steps, referred to as the processing pipelines. The pipeline used typically includes a tagger, a lemmatiser [4.2 Data Cleaning and Pre-processing], a parser, and an entity recogniser. Each pipeline component returns the processed DOC, which is then passed on to the next component. Spacy is built on the latest research and comes with pre-trained statistical models and word vectors, and currently supports tokenisation for 49+ languages. But when it comes to training the model for a specific task using Spacy, it requires the data to be pre-processed, i.e., with no noisy entries.

The pros of Spacy is that it is specially designed to perform limited NLP task, but cons come as Spacy cannot be used for modification and research purposes. As Spacy is integrated and opinionated, it tries to exclude the user by not asking them to choose between multiple algorithms that deliver same functionality and thus, increasing the overall time when compared to other models with huge datasets.

3.5 NLP

Natural Language Processing (NLP) is a field of study and application that looks at how computers can interpret and modify natural language text and speech in order to perform useful task. NLP also plays a growing role in enterprise solutions that help business operations to be more efficient, increase employee performance, and simplify mission-critical business processes.

NLP has made many tasks possible to be executed by the machine which were only humans were considered capable to perform such as spell check, keyword search, answering complex questions, text parsing, sentiment analysis, and mainly extracting featured information from different files.

3.6 BERT

BERT is an acronym for Bidirectional Encoder Representation from Transformers; it was launched by Google in 2019. It is a novel based approach of pre-training language representation that achieves cutting edge results on a wide range of Natural Language Processing tasks. It is built on Transformer, a deep learning model in which each output element is linked to every input element, and the weights between them are dynamically computed based upon their connection.

The main function of BERT is it can help different models understand language more as a human brain does. Compared to other models, BERT is different, rather than training the model using traditional way on the ordered sequence of words (left to right or combined right to left and left to right), it trains the model on the entire set of words in a query or sentence (Bidirectional training). The main idea behind BERT is that it allows language models to acquire word context from surrounding words rather than simply the one that comes before or after it.

BERT solves the problems in two phases:

1. Pre-train BERT to understand language:

BERT can be pre-trained on 2 different NLP tasks.

i. Masked Language Modelling (MLM)
ii. Next Sentence Prediction (NSP)

It learns language by training on the above unsupervised tasks simultaneously.

Masked Language Model (MLM) training objective is to hide a word or group of words in a sentence and then have the model predict what words have been hidden based on the surrounding words of the masked context.

Next Sentence Prediction (NSP) training is done with an objective to have the model predict whether two given sentences have a sequential, logical connection or whether their relationship is just random.

Figure 3-2 BERT Architecture below explains the working of BERT when working simultaneously on MLM and NSP.

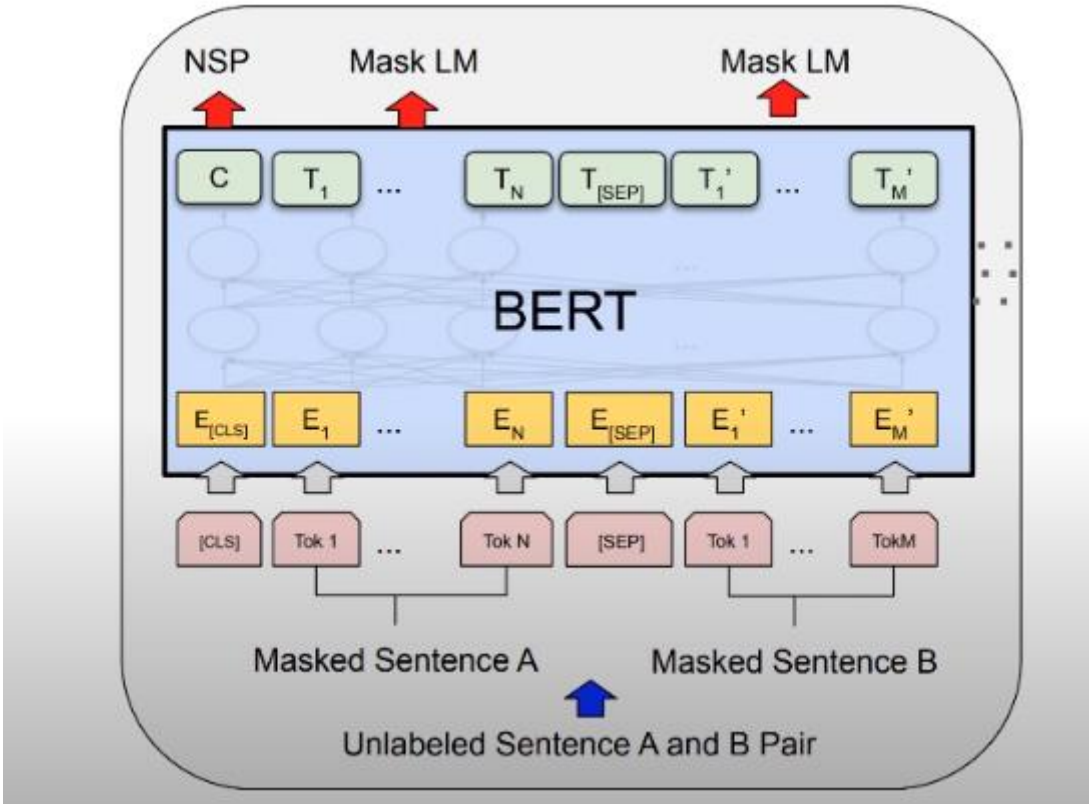

Figure 3-2 BERT Architecture

The sentence is passed to BERT as a list of tokens with some words masked. Each word is considered as a token. All the tokens are converted into embedding using pre-trained embedding. On the output side, 'C' is the binary output for NSP, and it tells if Sentence A is followed by Sentence B. Each Ts are word Vectors that correspond to the output of the MLM problem.

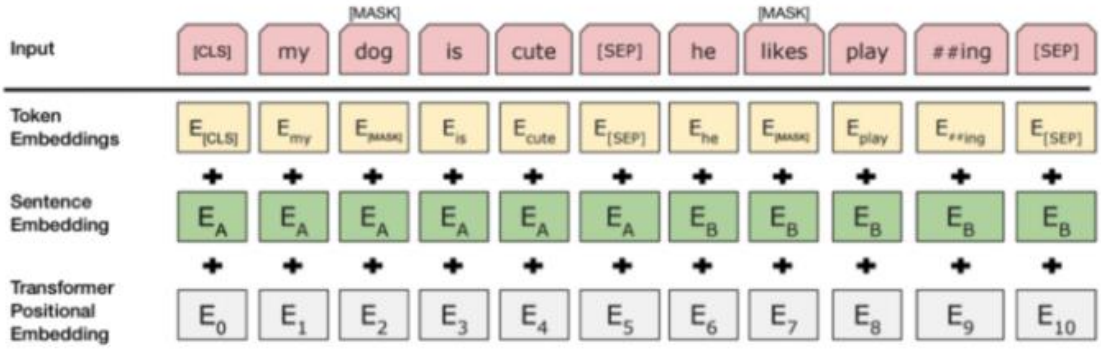

Figure 3-3 BERT Embedding Vector

The initial embedding is constructed using three vectors as shown above in Figure 3-3 BERT Embedding Vector.

Token embeddings are the pre-trained embeddings. It uses Word piece embedding which contains 30000 tokens.

The second is the Sentence Embedding. This is the number of a sentence which is encoded in the vector.

The position vector or the position embedding defines the position of a word within a sentence vector.

The embedding vector is obtained by combining the token embeddings, the sentence embedding and the positional embedding. Once the embedding vector has been defined, the next step is to procure the temporal ordering by feeding the BERT model with positional, embedding, and token vectors. The modelling process needs to preserve the temporal order during all the training iterations.

The segment embedding and position embedding are required for the temporal ordering as all these vectors are entered simultaneously into BERT. And the language modelling needs this order to be preserved.

Using all the vectors together BERT gets a good understanding of a Language in processing.

2. Fine-tune BERT to learn specific tasks:

When doing fine-tuning of the BERT model, it learns how to solve a specific set problem.

After pre-train, BERT could be moulded for very specific NLP tasks. All that needs to be done is replacing the fully connected output layers of the network with a fresh set of the output layer. After that supervised training could be performed using the selected dataset for that specific task. It does not take much time to train the BERT model, and it is fast as only the output layers are learned from the scratch and the rest of the model is just finely tuned.

“BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In this vanilla form, Transformer includes two separate mechanism – an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT’s goal is to generate a language model, only the encoder mechanism is necessary” (Horev, 2018).

3.7 Conditional Random Fields (CRF)

Conditional Random Fields classifier is a series modelling method. This checks if the features are independent of each other and think about the future observation while studying a new logic. This determines if the features are independent and takes into account future observations when studying a logic. CRF also incorporates the best features of both the Hidden Markov Model (HMM) and Maximum Entropy Markov Model (MEMM). When comparing the performance of CRF, HMM, and MEMM for entity recognition issues, CRF is comes out to be the best method. This is well explained in the section below.

- HMM: It is a sequence modelling algorithm for pattern-recognition and pattern-learning algorithm. It considers the future observations of the entities in order to develop a pattern, but it can also assume that they are all independent of one another.
- MEMM: It is also a sequencing modelling method. This algorithm does not consider that the entities are not dependent of each other and does not think about future investigation for studying the logic.

Mathematical Explanation of Conditional Random Fields:

The hidden state here is Y and the observed variable is X.

$$p(Y|X) = \frac{1}{Z(X)} \prod_{t=1}^{T} exp\left\{ \sum_{k=1}^{K} \theta_k f_k(Y_t, Y_t - 1, X_t) \right\}$$

where,

$\frac{1}{Z(X)}$ = Normalisation

$\theta_k f_k$= Weight

$(Y_t, Y_t - 1, X_t)$ = Feature

There are in total of two parts for the CRF Formula:

1. Normalisation: As it is seen that no probabilities are on the right side of the formula where we have the different features and sizes. Since, the expected output needs to be a probability and hence there is a want of normalisation. The normalisation constant Z(X) should become 1 as it is the sum of all feasible state series.
2. Features and Sizes: This part can be thought of as the logistic regression formula with sizes and the corresponding features. The estimated size is performed by maximum likelihood calculation and the features are set manually.

3.8 Entity Extraction and Filtration:

Entity extraction, also known as Named Entity Extraction (NER), allows machines to recognise and extract entities such as product names, locations, talents. Organisation names, and many more. With the rise in interest in Natural Language Processing, entity recognition and identification has seen a recent increase in popularity. An entity can generally be understood as a component of the document or text data which scientists find interesting to work on.

3.9 Proposed Method/Framework

The research methodology involves a new approach resume evaluation using BERT to find a model with the best accuracy for entity extraction and identification. The implementation of the research is discussed in the previous sections. Since the output is transmitted further to fine-tune the model for entity extraction and identification from distinct resumes, the proposed technique leverages the complete information incorporated in the pre-trained

BERT model. The dataset has been divided into two groups: test data and training data. The train data has been pre-processed according to the specifications of the proposed method i.e., BERT and other ML algorithm like Spacy and CRF. The dataset is cleaned of noise and made appropriate for training the model using a variety of approaches. This clean, pre-processed data is recorded in a data frame and then provided to the model that has already been trained. The BERT performs admirably as a feature extractor for text in this case.

The system's second half focuses on fine-tuning the model to obtain state-of-the-art outcomes for the proposed method. The fine-tuned BERT model will be used to extract and identify different entities from the resume. The dataset is first divided into a batch of 500 and further into train and test data. The research subject model i.e., BERT, trains on the training data/set with ten epochs where the dropout is set to 0.2. This stops the machine to remember the entities easily.

Multiple unstructured resumes will be passed to the fully trained model, and it will fetch the information from them. The extracted entities will be compared with the requirement of the job and the resumes will be ranked according to the information matched from the requirements.

BERT was compared with different models based on the resultant accuracy and performance matrices such as recall, precision and F1 score, and evaluation on a bulk of original resumes. An evaluation of performance of the BERT, Spacy, and CRF model to extract and identify featured or important information from resumes has been done. At the time of calculation, metrices such as accuracy, recall and precision gave the classifiers useful information about their classification abilities. When a classifier's accuracy is higher, it indicates that the classifier is more robust. The accurate outcomes predicted by the classifier are considered in the accuracy measurement. In the next chapter, Experiments and Results, the proposed approach is critically examined with the results of other machine learning models i.e., Spacy and CRF.

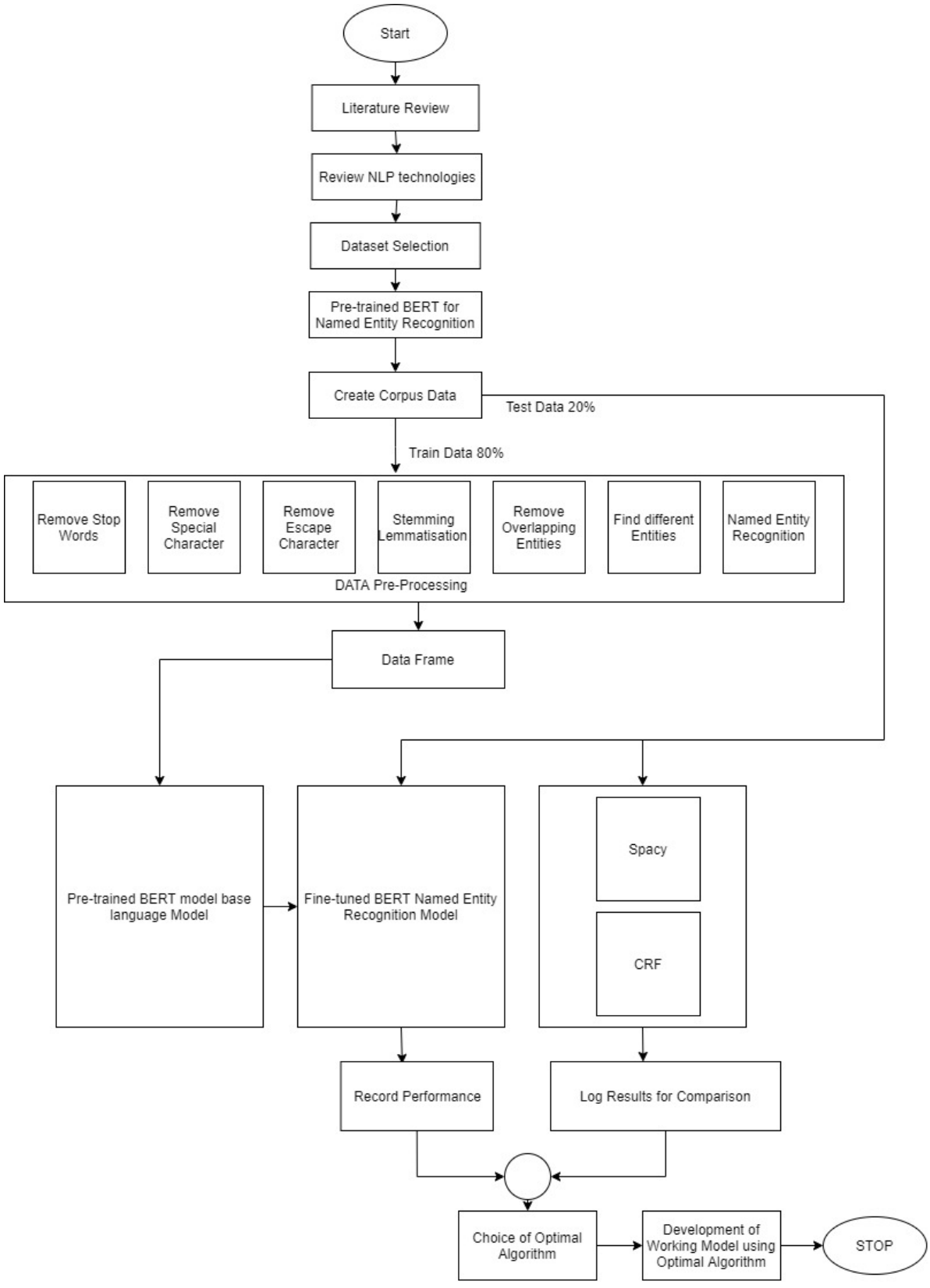

Figure 3-4 Project Flow Diagram

Chapter 4 – Experiments and Results

This chapter explains the implementation of Spacy, BERT, and CRF in python programming language while fully complying with the objectives defined in *Chapter 1*. The experiments demonstrated for Automated Resume Evaluation for a group of resumes from Kaggle to find the most suitable candidate for any job description have been presented with minute stages and information in the part below.

4.1 Data Collection

The data required for analysis using multiple Machine learning model should be enormous. The dataset used for the proposed model contains 220 resumes with information of the ann_labels with their start position, end position, and text. The collected dataset is further split into test and train data for the Named Entity Recognition task.

4.2 Data Cleaning and Pre-Processing

The data obtained from Kaggle contains information from different resumes of different formats. It also contains special character, extra new lines, null values, punctuations, and extra blank spaces between sentences, which are regarded as noises since they do not transmit adequate meaning when training the classification model. Cleaning and pre-processing data are critical stages as it is concerned with increasing the quality of the available data, hence the overall accuracy and productivity. If the model is trained using noisy data, it might result in less accuracy and efficiency of the model as seen in the study published by (Sonar, 2012). Following the procedures outlined below, a trustworthy and clean dataset was created for training.

- Remove Stop words:
  Stop words could be defined as words or character that do not specifically help in drawing a meaning from a sentence. Stop words are commonly used for the correctness of syntax for example the words, such as “an”, “the”, “and”, “on”, and many more, won’t help in understanding a skill from the resume. When it comes to entity extraction and identification, these terms must be excluded in automated resume evaluation because they are worthless in model training.
- Remove Special Characters:
  Special characters, such as different symbols, escape characters, such as \n, \t, and all non-ASCII characters, are all excluded using regular expression. The specialised function in the RE module aid pattern matching and filtering.
- Remove Punctuation:
  Punctuation has been omitted since it contributes more to the noise than to the context.
- Remove Leading and Trailing Spaces:
  Leading and training white spaces are removed from the entity spans. RE module was used to remove the matching pattern.

- Removing Repeated Entities:
  All the repeated entities such as different entities assigned to the same values are removed to get a cleaned dataset for the training of the model.
- Stemming and Lemmatisation:
  Stemming is performed to take the infected/derived words to their linguistic root, where lemmatisation takes the word to its original root that is Lemma. In the research, the advanced approach that is Lemmatisation has been performed.

4.3 Data Normalisation

The data “cleaning” and “pre-processing” is followed by the data normalisation. Data normalisation has to be performed and the extraction of all annotations has to be collected into a uniform sequence so that the data fed to the model is not vague and could be interpreted with only one meaning. The data frame with resume information and ann_labels information must be delivered in a format that would be suitable for identification. The implementation of the code to normalise the data has been given below:

```
def convert_goldparse(dataturks_JSON_FilePath):
    try:
        training_data = []
        lines = []
        with open(dataturks_JSON_FilePath, 'r', encoding="utf-8") as f:
            lines = f.readlines()

        for line in lines:
            data = json.loads(line)
            text = data['content'].replace("\n", " ")
            entities = []
            data_annotations = data['annotation']
            if data_annotations is not None:
                for annotation in data_annotations:
                    point = annotation['points'][0]
                    labels = annotation['label']
                    if not isinstance(labels, list):
                        labels = [labels]

                    for label in labels:
                        point_start = point['start']
                        point_end = point['end']
                        point_text = point['text']

                        lstrip_diff = len(point_text) - \
                            len(point_text.lstrip())
                        rstrip_diff = len(point_text) - \
                            len(point_text.rstrip())
                        if lstrip_diff != 0:
                            point_start = point_start + lstrip_diff
                        if rstrip_diff != 0:
                            point_end = point_end - rstrip_diff
                        entities.append((point_start, point_end + 1, label))
            training_data.append((text, {"entities": entities}))
        return training_data
    except Exception as e:
        logging.exception("Unable to process " +
                          dataturks_JSON_FilePath + "\n" + "error = " + str(e))
        return None
```

Figure 4-1 Implementation of Data Normalisation

4.4 Implementation of NER using Spacy

4.4.1 Data Pre-processing

Different functions are written to clean and pre-process the data according to the requirement of the Spacy model. The functions were developed to read all the entries of the dataset one by one. The JSON data is loaded to clear any noisy data which also includes special characters, null values, escape sequences, and repeating entities. After the dataset is cleaned, the processed data is passed to convert in a structured data frame using Regular Expression for further use. The data is divided into train and test data. Random function is imported to split the data in a random ratio.

4.4.2 Train Spacy Model

Spacy has a lot of in-built libraries developed to perform different NLP tasks. The function to train spacy creates the built-in pipeline components and adds them to the pipeline. A list of ann_labels is saved after processing all the resumes information from the dataset. The pipeline which are not pre-defined in Spacy are recorded and are disabled during training. The Spacy model is trained for the iteration of ten loops. The figure below shows the training iteration of the Spacy model:

```
Starting iteration 0
{'ner': 23356.18297057152}
Starting iteration 1
{'ner': 20344.348738635174}
Starting iteration 2
{'ner': 15360.129866825087}
Starting iteration 3
{'ner': 11570.559662660284}
Starting iteration 4
{'ner': 14676.677385685633}
Starting iteration 5
{'ner': 11314.993359747456}
Starting iteration 6
{'ner': 11169.14573702933}
Starting iteration 7
{'ner': 9914.609071474842}
Starting iteration 8
{'ner': 9100.245213174432}
Starting iteration 9
{'ner': 8912.13688363489}
```

Figure 4-2 Training of Spacy Model

After training the model, the trained model is fed with the test data and the data is parsed using GoldParse (an inbuilt spacy library). The performance metrics are recorded for further analysis and the accuracy of the implementation is recorded for further comparison.

The trained model is saved to the machine using Load command. Saving the model as a checkpoint help to process the model faster and saves time and resources of the system. The saved model is loaded back to the platform to test the model on a range of resumes sharing no common structure in a real environment.

4.4.3 Evaluation of Spacy on PDF file

PyMuPDF is installed using the pip command, which helps the model read PDF documents and convert all the context of the file into a text format without compromising the structure of the information. This converted data is then passed to the trained model for data pre-processing and then all the entity is extracted for the system to compare it with the skills and give a list of ranked Resumes.

4.4.4 Recording of the Classification Report

The classification report shown in Fig 4.4.1 Classification Report of Spacy Model is recorded and analysed further for the comparison between multiple ML models. See *"Appendix A Named Entity Recognition using Spacy"* for the full Spacy Code.

	precision	recall	f1-score	support
-	0.00	0.00	0.00	142
B-College Name	0.64	0.72	0.68	32
I-College Name	0.66	0.78	0.72	63
L-College Name	0.67	0.75	0.71	32
U-College Name	0.00	0.00	0.00	1
B-Companies worked at	0.49	0.83	0.62	30
I-Companies worked at	0.10	0.50	0.16	4
L-Companies worked at	0.47	0.80	0.59	30
U-Companies worked at	0.54	0.32	0.40	41
B-Degree	0.79	0.79	0.79	24
I-Degree	0.85	0.86	0.86	66
L-Degree	0.79	0.79	0.79	24
U-Degree	0.40	0.67	0.50	3
B-Designation	0.67	0.72	0.69	47
I-Designation	0.84	0.53	0.65	40
L-Designation	0.65	0.70	0.67	47
U-Designation	0.00	0.00	0.00	1
B-Email Address	1.00	1.00	1.00	7
L-Email Address	1.00	1.00	1.00	7
U-Email Address	0.67	1.00	0.80	10
U-Graduation Year	0.46	0.55	0.50	22
B-Location	0.00	0.00	0.00	3
L-Location	0.00	0.00	0.00	3
U-Location	0.48	0.66	0.55	32
B-Name	0.95	0.91	0.93	23
L-Name	0.95	0.91	0.93	23
O	0.94	0.98	0.96	12457
B-Skills	0.71	0.63	0.67	27
I-Skills	0.78	0.53	0.63	1022
L-Skills	0.62	0.56	0.59	27
U-Skills	0.00	0.00	0.00	2
B-Years of Experience	0.50	0.25	0.33	4
L-Years of Experience	0.50	0.25	0.33	4
U-Years of Experience	0.00	0.00	0.00	1
micro avg	0.92	0.92	0.92	14301
macro avg	0.53	0.56	0.53	14301
weighted avg	0.91	0.92	0.91	14301
samples avg	0.92	0.92	0.92	14301

Figure 4-3 Classification Report of Spacy Model

4.5 Implementation of NER using CRF

4.5.1 Data Pre-processing

The same methods used for the Spacy model for data cleaning and pre-processing are carried for the CRF model.

The function reads the dataset and loads it into JSON. After the dataset is converted to a JSON list, all the special characters, escape sequence, and extra white spaces are removed to have a cleaned dataset. CRF model requirement consists of the dataset with annotation to understand the connection between the entity and its surrounding sentence. Annotation is the process of tagging the words with the corresponding tag (BILOU). All the ann_labels in the dataset gets extracted with their starting position, ending position, and the annotation tag. All the entities in the dataset are fetched and checked for repetition. If the entities already belong in the list, it gets ignored, otherwise, the new entities are added into a list called training_data.

After training the model, a checkpoint is created to keep the log of previous iterations using the log parser function as shown in *Fig 4-4 Check Point in CRF*. This helps the CRF model to keep track of the training for future references.

```
trainer.logparser.last_iteration
```

```
{'active_features': 3043,
 'error_norm': 609.147043,
 'feature_norm': 44.838893,
 'linesearch_step': 1.0,
 'linesearch_trials': 1,
 'loss': 3243.317544,
 'num': 100,
 'scores': {},
 'time': 0.122}
```

Figure 4-4 Check Point in CRF

4.5.2 Train CRF model

Using Sklearn library confusion matrix, classification_report and accuracy_score models are imported to develop a classification report for further analysis and record accuracy of the model for further comparison with another machine learning model. A classification report for a list of BIO-encoded sequences is used. It computes token-level metrics and discards all the entities with “O” ann_labels.

4.5.3 Recording of the Classification Report

The classification report shown in Fig 4-5 Classification Report of CRF Model is recorded and analysed further for the comparison between multiple ML models. See *“Appendix B – Named Entity Recognition using CRF”* for the complete CRF code.

	precision	recall	f1-score	support
B-College Name	0.64	0.54	0.58	13
I-College Name	0.68	0.89	0.77	19
B-Companies worked at	0.74	0.41	0.53	34
I-Companies worked at	0.25	0.11	0.15	9
B-Degree	0.57	0.57	0.57	7
I-Degree	0.40	0.67	0.50	12
B-Designation	1.00	0.52	0.68	27
I-Designation	0.95	0.54	0.69	39
B-Email Address	0.80	0.89	0.84	9
I-Email Address	0.86	1.00	0.92	6
B-Graduation Year	0.67	0.29	0.40	14
B-Location	0.64	0.33	0.44	21
I-Location	0.00	0.00	0.00	2
B-Name	1.00	1.00	1.00	11
I-Name	1.00	0.92	0.96	12
O	0.94	0.97	0.95	2038
B-Skills	0.75	0.43	0.55	14
I-Skills	0.86	0.83	0.85	188
B-Years of Experience	0.00	0.00	0.00	2
I-Years of Experience	0.00	0.00	0.00	2
micro avg	0.92	0.92	0.92	2479
macro avg	0.64	0.55	0.57	2479
weighted avg	0.91	0.92	0.91	2479
samples avg	0.92	0.92	0.92	2479

Figure 4-5 Classification Report of CRF Model

4.6 Implementation of NER using BERT

4.6.1 Train and Fine-tune BERT model

A pre-trained BERT base uncased model has been extracted before to implement it on the normalised data.

Figure 4-5 Extraction of BERT base uncased

BERT, unlike any other Machine Learning model, features parallel processing of all the input tokens, extracted alongside the complete sequence of sentences from a data source in a single run.

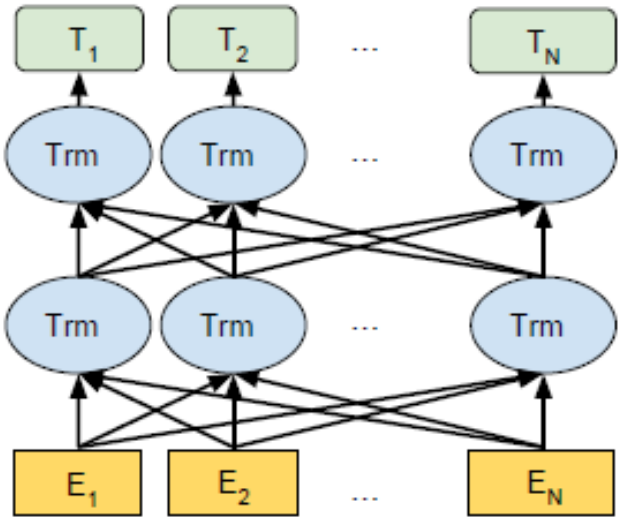

Figure 4-6 Pre-trained BERT model

Parallel extraction of the BERT's contextualised output vectors is depicted in the diagram above from the node T1 to the node TN. The BERT configuration used was pre-trained uncased BERT with 12 transformer arrays and 12 self-attention heads. The BERT output vectors were realised by further fine-tuning the pre-trained BERT. The BERT model has been implemented as an information extractor, and this extracted information is passed to the planned python function called the compare_filter_func, for comparing and ranking of resumes.

The input to the BERT is a JSON file "Resume.json". The JSON file initially, is not a valid JSON file but, has a collection of JSON data thus, making it necessary to pre-process the data in the following way:

1. JSONs are clearly separated.
2. Preceding and Trailing characters with do not contribute towards the meaning of the sentence. For ex. ";", "\n", "\t", whitespaces, etc.

The first step above to pre-process the data is achieved by using python file I/O package along with a JSON library and the second step is achieved by using Regular Expressions and once the data is cleaned of unwanted chars. The BERT implemented, was by making use of the "transformers" library alongside NumPy, and Pytorch library. BERT Tokeniser was used to tokenise the input text for the BERT. A batch size of 500 was used to train the BERT fine-tuned model. To achieve the results, the transformer was trained using previously split train data for 10 epochs. The figure below gives an insight into the training of the model:

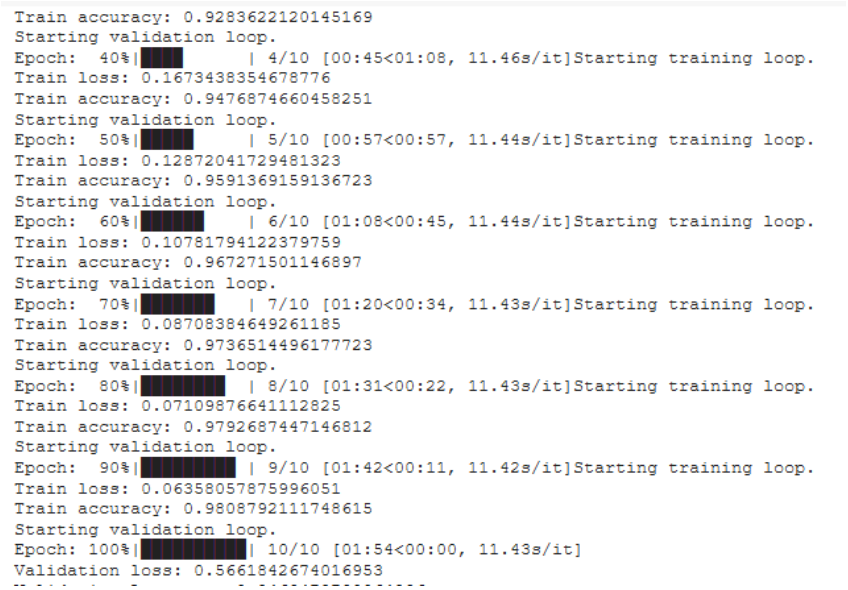

Figure 4-7 Training of BERT model

4.6.2 Recording of the Classification Report

The classification report shown in *Figure 4-8 Classification Table of BERT model* is recorded and analysed further with other models i.e., Spacy and CRF. The accuracy of the evaluation is being used for further comparison. See *"Appendix C – Named Entity Recognition using BERT"* for the complete BERT model code.

Classification Report:

	precision	recall	f1-score	support
Email Address	0.78	0.78	0.78	1142
Skills	0.41	0.76	0.53	902
Graduation Year	0.23	0.41	0.29	17
Designation	0.33	0.29	0.31	92
Location	0.55	0.77	0.64	53
Companies worked at	0.19	0.40	0.26	60
College Name	0.28	0.48	0.36	33
Name	0.86	0.93	0.89	41
Degree	0.53	0.51	0.52	35
Years of Experience	0.50	0.20	0.29	5
micro avg	0.53	0.73	0.62	2380
macro avg	0.59	0.73	0.64	2380

Figure 4-8 Classification Report of BERT model

4.6.3 Implementation of Compare_filter_func

The skills required for the job description are entered in a variable list. The resumes that need to be check are then passed in a FOR loop for our BERT model to execute on them one by one. All the information is extracted, and it is compared with the requirement. For every skill matched the CV is ranked and the output of the function is a list of ranked resumes with their score.

4.8 Training ML models multiple times for Better Precision

The number of iterations were changed to record the effect of it on accuracy. It was considered that a greater number of training iterations result in better accuracy. The results collected for different ML models i.e., Spacy, BERT, and CRF are short to answer one of our research questions.

4.9 Results and Discussion

In the preceding chapters, the methodology and description of the experiments conducted for the proposed research were thoroughly discussed. This section of the study compares the results of the proposed techniques BERT with the results of existing machine learning algorithm such as Spacy and CRF.

Firstly, the dataset is collected from Kaggle as stated in the preceding section. There are 220 resumes and the location of all the featured information in those resumes.

The data is extracted by reading the dataset line by line and then after using multiple functions, the data was cleaned and pre-processed. The pre-processed data is saved to a list and all the extra white spaces were truncated. The normalised data is then split into test and train data sets to implement the execution of the entity extraction and classification model.

The BERT, Spacy, and CRF machine learning models are applied for the entity extraction and classification on the normalised dataset achieved from the above. The result of the classifiers is visualised, and the accuracy has been assessed to make a comparison.

Classification Model	Accuracy	Precision	Recall	F1_Score
NER using Spacy	82%	88%	82%	85%
NER using CRF	91.60%	64%	55%	57%
NER using BERT	94.72%	62%	79%	69%

Table 4-1 Comparison of Accuracy

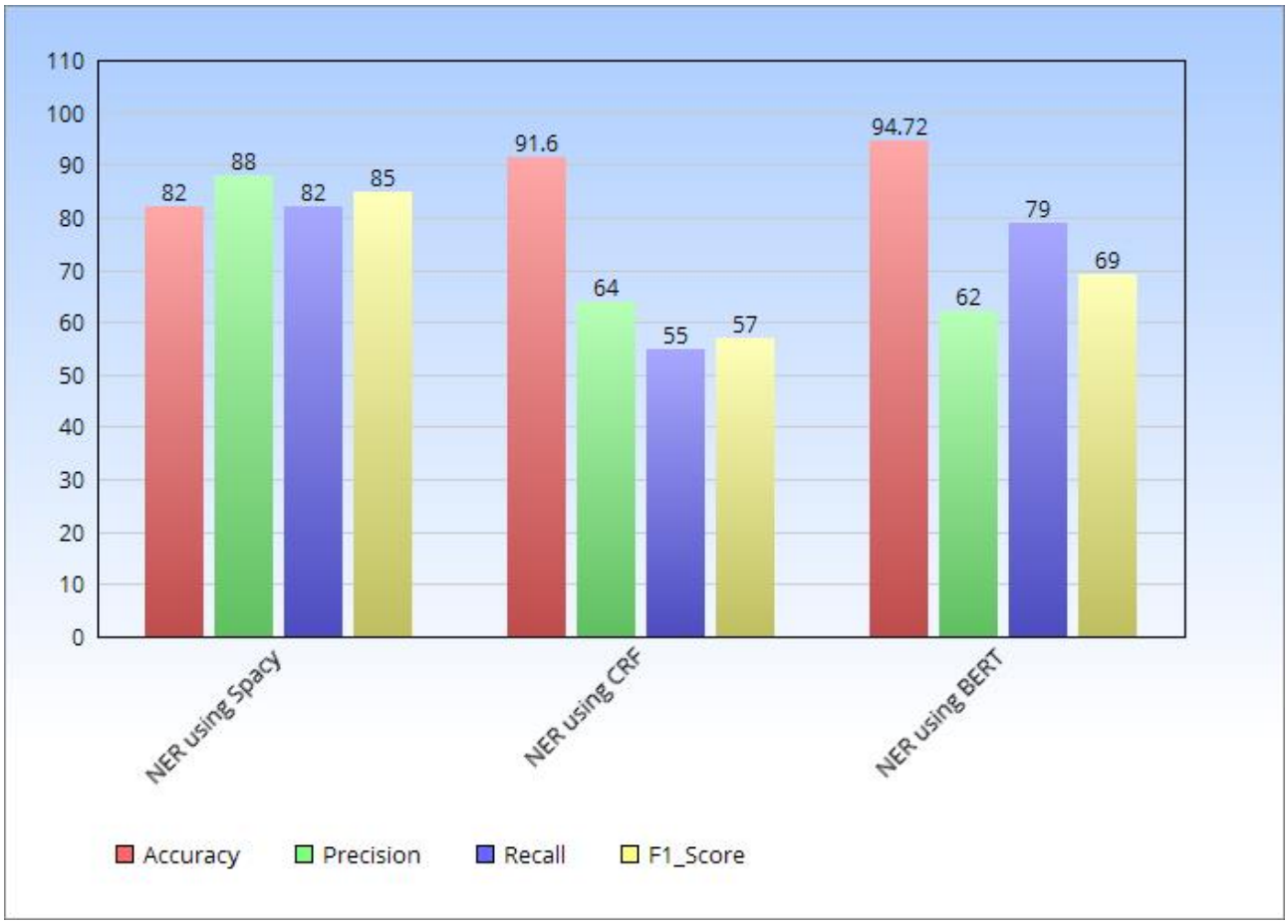

Figure 4-10 Comparison of Accuracy

The Recall, Precision, and F1_score is calculated for each entity individually and all the values are averaged to find the values for the model.

NER using Spacy:

The values lying under different ann_labels such as college name, companies worked at, degree, and many more are tagged using BILUO to provide annotation to the model with the text information. Spacy's accuracy, at 82%, is considered moderate. It also has a moderate recall and precision rate. Spacy's overall performance is likewise good, as seen by its F1_score.

NER using CRF:

Conditional Random Field Classifier shows an excellent accuracy of 91.60% and precision are good among the comparison. F1_score of CRF indicates how exact its categorisation is.

NER using BERT:

The BERT has shown the highest accuracy among all the classifiers with a training accuracy of 94.27%. With the highest accuracy and a moderate F1_score, BERT seems to show that the BERT classification is good taking Precision of 62% and recall of 79% into consideration.

As predicted above, BERT has the highest accuracy metrics than Spacy and CRF. Spacy shows a moderate performance when the precision, recall, and F1_score is compared with other classifiers.

After running the tests on all the models using an unstructured resume file for entity extraction and classification. BERT transformer outperformed the performance of the Spacy and the CRF model.

The output containing extracted entities from the document seems to be the best using the BERT model. With the highest accuracy and best evaluation output using a real unstructured resume, BERT fits to be the best model among all the three models (Spacy, BERT, and CRF).

The final model: Automated Resume Evaluation System is developed using BERT as the classifier because of its robustness. The final output experiment will be able to accept different numbers of resumes and will rank them according to the information matching with the job description.

Chapter 5 – Conclusion and Future Work

5.1 Conclusion

Apart from the accuracy during entity extraction and classification, the comparison and discussion in the preceding chapter provided insight on several performance tests. This dissertation is completely based on the dataset taken from Kaggle containing 220 resumes and the location of all the featured information in the dataset. The suggested architecture was designed, implemented, and compared using BERT to enhance entity extraction and identification and resilience in the NLP task: Automated Resume Evaluation.

Furthermore, the BERT model's finding was shown to outperform the results of Spacy and CRF in real-time resume evaluation. Hence the performance of the model may not be completely dependent on the results figure but also their advantage in real-time.

Moreover, in section something, it was cleared that it is not necessary that training the model multiple times will improve the performance of the model, as it depends on the number of iterations for training models and the type of Machine Learning Model.

From the comparison result in section something, the following things were observed:

For Spacy: Increasing or decreasing the number of iterations does not make any great difference in the accuracy of the model suggesting that there exists a saturation state from where the number of iterations would not increase the accuracy of the model.

For CRF: The accuracy of the model changes with the number of iterations for the CRF model. If the model was trained for less than or equal to 40 iterations, the accuracy took a severe hit and for iteration=100 the accuracy reached an astonishing 91%.

For BERT: From the result, BERT gives the accuracy in a constant range when the epoch is set between 10 to 20. But as soon the epoch is increased to 30. The accuracy of the model seemed to be improved but when the model was tested on real unstructured resumes, the entity extracted came out to be not precise. This suggests that the corpus used did not have enough technical skills listed and is not a limitation of the BERT model.

For answering the last research question, Bi-directional Entity Recognition for Transformer outperformed the other machine model in terms of accuracy and precision when worked in a real environment with multiple unstructured resumes.

5.2 Future Work

The experiment demonstrated the Automated resume evaluation system by the successful integration of the BERT algorithm. The research was centred around the entity extraction, classification, comparison, and filtration of resumes received for a specific job description. In near future, the following improvements could be made in the research:

- The resume evaluation system focuses to compare on skills required for the job and does not take work experience and certification into consideration. This creates a scope for a model with a better understanding of how relevant the candidate's skills are at present.
- The entity extracted during the evaluation of resumes for a specific job description could be stored in the dataset with user concern to train the model better.
- There's some area for improvement, which might be addressed by expanding the input size dataset with more new resumes for the model to train on.

REFERENCE

1. Falk, G., 2020. *Unemployment Rates During the COVID-19 Pandemic:*. Congressional Research Service.
2. Zubeda, J.A.A., 2016. " Resume Ranking using NLP and Machine Learning.
3. Nimbekar, R., Patil, Y., Prabhu, R. and Mulla, S., 2019, December. Automated Resume Evaluation System using NLP. In *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)* (pp. 1-4). IEEE.
4. Dragoni, M., Villata, S., Rizzi, W. and Governatori, G., 2016, December. Combining NLP approaches for rule extraction from legal documents. In *1st Workshop on MIning and REasoning with Legal texts (MIREL 2016)*.
5. Amin, S., Jayakar, N., Kiruthika, M. and Gurjar, A., 2019. Best Fit Resume Predictor. *International Research Journal of Engineering and Technology*, *6*(8), pp.813-820
6. Han, X. and Wang, L., 2020. A novel document-level relation extraction method based on BERT and entity information. *IEEE Access*, *8*, pp.96912-96919.
7. Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
8. Bhatia, V., Rawat, P., Kumar, A. and Shah, R.R., 2019. End-to-End Resume Parsing and Finding Candidates for a Job Description using BERT. *arXiv preprint arXiv:1910.03089*.
9. Sonar, S. and Bankar, B., 2012. Resume parsing with named entity clustering algorithm. *paper, SVPM College of Engineering Baramati, Maharashtra, India*.
10. Nongmeikapam, K., Shangkhunem, T., Chanu, N.M., Singh, L.N., Salam, B. and Bandyopadhyay, S., 2011, March. Crf based name entity recognition (ner) in manipuri: A highly agglutinative indian language. In *2011 2nd National Conference on Emerging Trends and Applications in Computer Science* (pp. 1-6). IEEE.
11. Ayishathahira, C.H., Sreejith, C. and Raseek, C., 2018, July. Combination of neural networks and conditional random fields for efficient resume parsing. In *2018 International CET Conference on Control, Communication, and Computing (IC4)* (pp. 388-393). IEEE.
12. Regulation, G.D.P., 2018. General data protection regulation (GDPR). *Intersoft Consulting, Accessed in October*, *24*(1).
13. *Pretrained models*. (2019). Retrieved from https://huggingface.co/transformers/pretrained_models.html
14. Regulation, G. 2. (2018). General data protection regulation (GDPR).
15. Ramshaw, L. a. (1995). Text Chunking using Transformation-Based Learning. In L. a. Ramshaw, *Third Workshop on Very Large Corpora.*
16. Cloud, G. (n.d.). *Cloud Tensor Processing Units (TPUs)* . Retrieved from Cloud Tensor Processing Units (TPUs) : https://cloud.google.com/tpu/docs/tpus

17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

Appendices

Code

Appendix A – NER using Spacy

Processing Indexes

Same code has been used for CRF in Appendix B

```
# Formatting functions for JSON
def convert_dataturks_to_spacy(json_filepath):
    training_set = []
    file_lines = []
    with open(json_filepath, 'r', encoding="utf-8") as file:
        lines = file.readlines()

    for line in lines:
        if line=="":
            print("empty line")
            continue
        data_dict = json.loads(line)
        text = data_dict['content'].replace("\n", " ")
        entities = []
        annotations = data_dict['annotation']
        if annotations is not None:
            for annotation in annotations:
                #only a single point in text annotation.
                point = annotation['points'][0]
                labels = annotation['label']
                # handle both list of labels or a single label.
                if not isinstance(labels, list):
                    labels = [labels]

                for label in labels:
                    start_point = point['start']
                    end_point = point['end']
                    text_point = point['text']

                    lstrip_diff = len(text_point) - len(text_point.lstrip())
                    rstrip_diff = len(text_point) - len(text_point.rstrip())
                    if lstrip_diff != 0:
                        start_point = start_point + lstrip_diff
                    if rstrip_diff != 0:
                        end_point = end_point - rstrip_diff
                    entities.append((start_point, end_point + 1 , label))
        training_set.append((text, {"entities" : entities}))
```

```
    return training_set

def trim_entity_spans(data: list) -> list:
    invalid_span_tokens = re.compile(r'\s')

    cleaned_data = []
    for text, annotations in data:
        entities = annotations['entities']
        valid_entities = []
        for start, end, label in entities:
            valid_start = start
            valid_end = end
            while valid_start < len(text) and invalid_span_tokens.match(
                    text[valid_start]):
                valid_start += 1
            while valid_end > 1 and invalid_span_tokens.match(
                    text[valid_end - 1]):
                valid_end -= 1
            valid_entities.append([valid_start, valid_end, label])
        cleaned_data.append([text, {'entities': valid_entities}])
    return cleaned_data
```

Cleaning Entities

Same code has been used for CRF in Appendix B

```
 def clean_entities(training_data):

     clean_data = []
     for text, annotation in training_data:

         entities = annotation.get('entities')
         entities_copy = entities.copy()

         # append entity only if it is longer than its overlapping entity
         i = 0
         for entity in entities_copy:
             j = 0
             for overlapping_entity in entities_copy:
                 # Skip self
                 if i != j:
                     e_start, e_end, oe_start, oe_end = entity[0], entity[1]
, overlapping_entity[0], overlapping_entity[1]
                     # Delete any entity that overlaps, keep if longer
                     if ((e_start >= oe_start and e_start <= oe_end) \
                     or (e_end <= oe_end and e_end >= oe_start)) \
                     and ((e_end - e_start) <= (oe_end - oe_start)):
                         entities.remove(entity)
                 j += 1
```

```
            i += 1
        clean_data.append((text, {'entities': entities}))

    return clean_data
data = clean_entities(data)
```

Training of Spacy Model

```
def train_spacy():
    # Open-source code available from Spacy Library dev pages
    nlp = spacy.blank('en')  # create blank Language class
    # create the built-in pipeline components and add them to the pipeline
    if 'ner' not in nlp.pipe_names:
        ner = nlp.create_pipe('ner')    # works for built-
ins that are registered with spaCy
        nlp.add_pipe(ner, last=True)

    # adding labels
    for _, annotations in training_data:
        for ent in annotations.get("entities"):
            ner.add_label(ent[2])

    # get names of other pipes to disable them during training
    other_pipes = [pipe for pipe in nlp.pipe_names if pipe != 'ner']
    with nlp.disable_pipes(*other_pipes):  # only train NER
        optimizer = nlp.begin_training()
        for itn in range(10):
            print("Starting iteration " + str(itn))
            random.shuffle(training_data)
            losses = {}
            for text, annotations in training_data:
                nlp.update(
                    [text],  # batch of texts
                    [annotations],  # batch of annotations
                    drop=0.2,  # dropout - makes it harder for model to memo
rise data
                    sgd=optimizer,  # callable to update weights
                    losses=losses)
            print(losses)
    return nlp

def doc_to_bilou(nlp, text):

    doc = nlp(text)
    tokens = [(tok.text, tok.idx, tok.ent_type_) for tok in doc]
    entities = []
    for entity, group in groupby(tokens, key=lambda t: t[-1]):
        if not entity:
```

```
            continue
        group = list(group)
        _, start, _ = group[0]
        word, last, _ = group[-1]
        end = last + len(word)

        entities.append((
                start,
                end,
                entity
            ))

    gold = GoldParse(nlp(text), entities = entities)
    pred_ents = gold.ner

    return pred_ents

y_test = []
y_pred = []

for text, annots in testing_data:

    gold = GoldParse(nlp.make_doc(text), entities = annots.get("entities"))
    ents = gold.ner
    pred_ents = doc_to_bilou(nlp, text)

    y_test.append(ents)
    y_pred.append(pred_ents)

# Classification report.

# Same for CRF in Appendix B

def ner_report(y_true, y_pred):
    label_bin = LabelBinarizer()
    y_pred_total = label_bin.transform(list(chain.from_iterable(y_pred)))
    y_true_total = label_bin.fit_transform(list(chain.from_iterable(y_true))
)

    tags = set(lb.classes_)
    tags = sorted(tags, key=lambda tag: tag.split('-', 1)[::-1])
    class_pointers = {cls: idx for idx, cls in enumerate(label_bin.classes_)
}

    return classification_report(
        y_true_total,
        y_pred_total,
        labels = [class_pointers[cls] for cls in tags],
        target_names = tagset
```

```
    ), accuracy_score(y_true_total, y_pred_total)

report, accuracy = ner_report(y_test, y_pred)
print(report,accuracy)
```

Testing of Spacy Model

```
document = nlp_model(fetched_txt)
for entry in document.ents:
    print(f'{entry.label_.upper():{30}}- {entry.text}')

# Output of File
NAME                           - Alice Clark
LOCATION                       - Delhi
EMAIL ADDRESS                  - •
COMPANIES WORKED AT            - Microsoft
DESIGNATION                    - Software Engineer
COMPANIES WORKED AT            - Microsoft
LOCATION                       - Bangalore
COMPANIES WORKED AT            - Microsoft
COMPANIES WORKED AT            - Microsoft
COMPANIES WORKED AT            - Microsoft
COMPANIES WORKED AT            - Microsoft
COMPANIES WORKED AT            - Microsoft
LOCATION                       - Store
DEGREE                         - EDUCATION
COLLEGE NAME                   - Indian Institute of Technology - Mumbai
GRADUATION YEAR                - 2001
SKILLS                         - Machine Learning, Natural Language
Processing, and Big Data Handling
```

Appendix B – NER using CRF

```
# Defining BILOU Tags

def bilou_tags(data):

    docs  = []
    annots = []
    nlp = English()
    for text, annotations in data:
        offsets = annotations["entities"]
        doc = nlp(text)
        tags = biluo_tags_from_offsets(doc, offsets)
        for i in range(len(tags)):
            if tags[i].startswith("U"):
                tags[i] = "B" + tags[i][1:]
            elif tags[i].startswith("L"):
                tags[i] = "I" + tags[i][1:]
            if not (doc[i].text.isalnum() or len(doc[i].text) > 1):
                tags[i] = "O"
        docs.append([token.text for token in doc])
```

```
        annots.append(tags)

    df_data = pd.DataFrame({'docs': docs, 'annots': annots})

    return df_data

df_data = bilou_tags(data)

# Entity extraction
def feature_extractor(data_pack, i):
    syllable = data_pack[i][0]
    tag_pos_vector = data_pack[i][1]

    features = {
        'bias': 1.0,
        'syllable.lower()': syllable.lower(),
        'syllable[-3:]': syllable[-3:],
        'syllable[-2:]': syllable[-2:],
        'syllable.isupper()': syllable.isupper(),
        'syllable.istitle()': syllable.istitle(),
        'syllable.isdigit()': syllable.isdigit(),
        'tag_pos_vector': tag_pos_vector,
        'tag_pos_vector[:2]': tag_pos_vector[:2]
    }
    if i > 0:
        syllable1 = data_pack[i-1][0]
        tag_pos_vector1 = data_pack[i-1][1]
        features.update({
            '-1:syllable.lower()': syllable1.lower(),
            '-1:syllable.istitle()': syllable1.istitle(),
            '-1:syllable.isupper()': syllable1.isupper(),
            '-1:tag_pos_vector': tag_pos_vector1,
            '-1:tag_pos_vector[:2]': tag_pos_vector1[:2]
        })
    else:
        features['BOS'] = True
    if i < len(data_pack)-1:
        syllable1 = data_pack[i+1][0]
        tag_pos_vector1 = data_pack[i+1][1]
        features.update({
            '+1:syllable.lower()': syllable1.lower(),
            '+1:syllable.istitle()': syllable1.istitle(),
            '+1:syllable.isupper()': syllable1.isupper(),
            '+1:tag_pos_vector': tag_pos_vector1,
            '+1:tag_pos_vector[:2]': tag_pos_vector1[:2]
        })
    else:
        features['EOS'] = True
    return features
```

Appendix C – NER using BERT

Label Extraction

```
def get_label(offset, labels):
    if offset[0] == 0 and offset[1] == 0:
        return 'O'
    for label in labels:
        if offset[1] >= label[0] and offset[0] <= label[1]:
            return label[2]
    return 'O'
tags_vals = ["UNKNOWN", "O", "Name", "Degree", "Skills", "College Name", "Em
ail Address",
             "Designation", "Companies worked at", "Graduation Year", "Years
 of Experience", "Location"]
```

Data extraction from Resumes

```
def processing_resume(data, tokenizer, tag2idx, max_len, is_test=False):
    tok = tokenizer.encode_plus(
        data[0], max_length=max_len, return_offsets_mapping=True)
    curr_sent = {'orig_labels': [], 'labels': []}

    padding_length = max_len - len(tok['input_ids'])

    if not is_test:
        labels = data[1]['entities']
        labels.reverse()
        for off in tok['offset_mapping']:
            label = get_label(off, labels)
            curr_sent['orig_labels'].append(label)
            curr_sent['labels'].append(tag2idx[label])
        curr_sent['labels'] = curr_sent['labels'] + ([0] * padding_length)

    curr_sent['input_ids'] = tok['input_ids'] + ([0] * padding_length)
    curr_sent['token_type_ids'] = tok['token_type_ids'] + \
        ([0] * padding_length)
    curr_sent['attention_mask'] = tok['attention_mask'] + \
        ([0] * padding_length)
    return curr_sent
```

Retrieve tokens from dataset

```
def get_special_tokens(tokenizer, tag2idx):
    vocab = tokenizer.get_vocab()
    pad_tok = vocab["[PAD]"]
    sep_tok = vocab["[SEP]"]
    cls_tok = vocab["[CLS]"]
```

```
    o_lab = tag2idx["O"]

    return pad_tok, sep_tok, cls_tok, o_lab
```

Confusion matrics generator

```
def annot_confusion_matrix(valid_tags, pred_tags):
    """Creating annotated confusion matrix by adding labels, annotations and
 formatting to sklearn's `confusion_matrix`."""
    header = sorted(list(set(valid_tags + pred_tags)))
    matrix = confusion_matrix(valid_tags, pred_tags, labels=header)
    mat_formatted = [header[i] + "\t\t\t" + str(row) for i, row in enumerate
(matrix)]
    content = "\t" + " ".join(header) + "\n" + "\n".join(mat_formatted)
    return content

def flat_accuracy(valid_tags, pred_tags):
    return (np.array(valid_tags) == np.array(pred_tags)).mean()
```

Training BERT

```
def train_and_val_model(
  model,
  tokenizer,
  optimizer,
  epochs,
  idx2tag,
  tag2idx,
  max_grad_norm,
  device,
  train_dataloader,
  valid_dataloader
):


  pad_tok, sep_tok, cls_tok, o_lab = get_special_tokens(tokenizer, tag2idx)


  epoch = 0
  for _ in trange(epochs, desc="Epoch"):
    epoch += 1
```

```
# Training Iteration
print("Starting training iteration.")
model.train()
tr_loss, tr_accuracy = 0, 0
train_eg, train_levels = 0, 0
predictions, tr_labels = [], []

for step, batch in enumerate(train_dataloader):
    batch_id, input_mask, batch_labels = batch['input_ids'], batch['attention_mask'], batch['labels']
    batch_id, input_mask, batch_labels = batch_id.to(
        device), input_mask.to(device), batch_labels.to(device)

    # Forward pass
    outputs = model(
        batch_id,
        token_type_ids=None,
        attention_mask=input_mask,
        labels=batch_labels,
    )
    loss, tr_logits = outputs[:2]

    # Backward pass
    loss.backward()

    # Compute train loss
    tr_loss += loss.item()
    train_eg += batch_id.size(0)
    train_levels += 1

    # Subset out unwanted predictions on CLS/PAD/SEP tokens
```

```
        preds_mask = (
            (batch_id != cls_tok)
            & (batch_id != pad_tok)
            & (batch_id != sep_tok)
        )

        tr_logits = tr_logits.cpu().detach().numpy()
        tr_label_ids = torch.masked_select(batch_labels, (preds_mask == 1))
        preds_mask = preds_mask.cpu().detach().numpy()
        tr_batch_preds = np.argmax(tr_logits[preds_mask.squeeze()], axis=1)
        tr_batch_labels = tr_label_ids.to("cpu").numpy()
        predictions.extend(tr_batch_preds)
        tr_labels.extend(tr_batch_labels)

        # Compute training accuracy
        tmp_tr_accuracy = flat_accuracy(tr_batch_labels, tr_batch_preds)
        tr_accuracy += tmp_tr_accuracy

        # Gradient clipping
        torch.nn.utils.clip_grad_norm_(
            parameters=model.parameters(), max_norm=max_grad_norm
        )

        # Update parameters
        optimizer.step()
        model.zero_grad()

    tr_loss = tr_loss / train_levels
    tr_accuracy = tr_accuracy / train_levels

    # Print training loss and accuracy per epoch
```

```
print(f"Train loss: {tr_loss}")
print(f"Train accuracy: {tr_accuracy}")

"""
Validation loop
"""
print("Starting validation loop.")

model.eval()
eval_loss, eval_accuracy = 0, 0
nb_eval_steps, nb_eval_examples = 0, 0
predictions, true_labels = [], []

for batch in valid_dataloader:

  batch_id, input_mask, batch_labels = batch['input_ids'], batch['attention_mask'], batch['labels']
  batch_id, input_mask, batch_labels = batch_id.to(
    device), input_mask.to(device), batch_labels.to(device)

  with torch.no_grad():
    outputs = model(
      batch_id,
      token_type_ids=None,
      attention_mask=input_mask,
      labels=batch_labels,
    )
    tmp_eval_loss, logits = outputs[:2]

  # Subset out unwanted predictions on CLS/PAD/SEP tokens
  preds_mask = (
    (batch_id != cls_tok)
```

```
            & (batch_id != pad_tok)
            & (batch_id != sep_tok)
        )

        logits = logits.cpu().detach().numpy()
        label_ids = torch.masked_select(batch_labels, (preds_mask == 1))
        preds_mask = preds_mask.cpu().detach().numpy()
        val_batch_preds = np.argmax(logits[preds_mask.squeeze()], axis=1)
        val_batch_labels = label_ids.to("cpu").numpy()
        predictions.extend(val_batch_preds)
        true_labels.extend(val_batch_labels)

        tmp_eval_accuracy = flat_accuracy(
            val_batch_labels, val_batch_preds)

        eval_loss += tmp_eval_loss.mean().item()
        eval_accuracy += tmp_eval_accuracy

        nb_eval_examples += batch_id.size(0)
        nb_eval_steps += 1

    # Evaluate loss, acc, conf. matrix, and class. report on devset
    pred_tags = [idx2tag[i] for i in predictions]
    valid_tags = [idx2tag[i] for i in true_labels]
    cl_report = classification_report(valid_tags, pred_tags)
    conf_mat = annot_confusion_matrix(valid_tags, pred_tags)
    eval_loss = eval_loss / nb_eval_steps
    eval_accuracy = eval_accuracy / nb_eval_steps

# Report metrics
print(f"Validation loss: {eval_loss}")
```

```
    print(f"Validation Accuracy: {eval_accuracy}")
    print(f"Classification Report:\n {cl_report}")
    print(f"Confusion Matrix:\n {conf_mat}")

# Evaluation of BERT
output_path = "./drive/MyDrive/demo"
MAX_LEN = 512
EPOCHS = 10
MAX_GRAD_NORM = 1.0
MODEL_NAME = 'bert-base-uncased'
TOKENIZER = BertTokenizerFast('./drive/MyDrive/vocab/vocab.txt', lowercase=True)
MACHINE = torch.MACHINE("cuda" if torch.cuda.is_available() else "cpu")

data = trim_entity_spans(convert_goldparse('./drive/MyDrive/data/Resumes.json'))

total = len(data)
training_set, val_data = data[:180], data[180:]

training_set = ResumeDataset(training_set, TOKENIZER, tag2idx, MAX_LEN)
val_d = ResumeDataset(val_data, TOKENIZER, tag2idx, MAX_LEN)

train_sampler = RandomSampler(training_set)
training_setl = DataLoader(training_set, sampler=train_sampler, batch_size=8)

val_dl = DataLoader(val_d, batch_size=4)

model = BertForTokenClassification.from_pretrained(
    MODEL_NAME, num_labels=len(tag2idx))
model.to(MACHINE)
optimizer_grouped_parameters = get_hyperparameters(model, True)
optimizer = Adam(optimizer_grouped_parameters, lr=3e-5)
```

```
train_and_val_model(
   model,
   TOKENIZER,
   optimizer,
   EPOCHS,
   idx2tag,
   tag2idx,
   MAX_GRAD_NORM,
   MACHINE,
   training_setl,
   val_dl
)

torch.save(
   {
      "model_state_dict": model.state_dict()
   },
   f'{output_path}/model-state.bin',
)

# Testing on a demo resume
MAX_LEN = 500
NUM_LABELS = 12
MACHINE = torch.MACHINE("cuda" if torch.cuda.is_available() else "cpu")
MODEL_PATH = 'bert-base-uncased'
STATE_DICT = torch.load("./drive/MyDrive/demo/model-state.bin", map_location=MACHINE)
TOKENIZER = BertTokenizerFast("./drive/MyDrive/vocab/vocab.txt", lowercase=True)

model = BertForTokenClassification.from_pretrained(
   'bert-base-uncased', state_dict=STATE_DICT['model_state_dict'], num_labels=NUM_LABELS)
```

```
model.to(MACHINE)


def predict_api():
   data = open("./drive/MyDrive/demo/a.pdf", "rb")
   data = data.read()
   data = io.BytesIO(data)
   resume_text = preprocess_data(data)
   entities = predict(model, TOKENIZER, idx2tag,
               MACHINE, resume_text, MAX_LEN)
   output = dict()
   output["entities"] = entities
   json_object = json.dumps(output, indent = 4)
   print(json_object)


predict_api()


# Output BERT model


    "entities": [
        {
            "entity": "Name",
            "start": 3,
            "end": 19,
            "text": "Ayush Srivastava"
        },
        {
            "entity": "Designation",
            "start": 22,
            "end": 35,
            "text": "Web Developer"
        },
        {
            "entity": "Degree",
            "start": 50,
            "end": 56,
            "text": "B.Tech"
        },
        {
            "entity": "Years of Experience",
            "start": 72,
            "end": 74,
            "text": "3+"
```

```
},
{
    "entity": "Years of Experience",
    "start": 76,
    "end": 81,
    "text": "years"
},
{
    "entity": "College Name",
    "start": 204,
    "end": 207,
    "text": "JSS"
},
{
    "entity": "Designation",
    "start": 413,
    "end": 420,
    "text": "Analyst"
},
{
    "entity": "Designation",
    "start": 582,
    "end": 599,
    "text": "Software Engineer"
},
{
    "entity": "College Name",
    "start": 937,
    "end": 944,
    "text": "JSSATE,"
},
{
    "entity": "College Name",
    "start": 946,
    "end": 951,
    "text": "Noida"
},
{
    "entity": "Degree",
    "start": 955,
    "end": 957,
    "text": "B."
},
{
    "entity": "College Name",
    "start": 957,
    "end": 961,
    "text": "Tech"
},
{
    "entity": "Graduation Year",
    "start": 964,
    "end": 968,
    "text": "2016"
},
{
    "entity": "Skills",
    "start": 1054,
```

```
    "end": 1069,
    "text": "Data Structures"
},
{
    "entity": "Skills",
    "start": 1071,
    "end": 1072,
    "text": "\u25cb"
},
{
    "entity": "Skills",
    "start": 1074,
    "end": 1088,
    "text": "SQL based DBMS"
},
{
    "entity": "Skills",
    "start": 1094,
    "end": 1109,
    "text": "Turing Machines"
},
{
    "entity": "Skills",
    "start": 1114,
    "end": 1116,
    "text": "OS"
},
{
    "entity": "Skills",
    "start": 1168,
    "end": 1185,
    "text": "C/C++ Programming"
},
{
    "entity": "Skills",
    "start": 1190,
    "end": 1199,
    "text": "Designing"
},
{
    "entity": "Skills",
    "start": 1201,
    "end": 1204,
    "text": "UI/"
},
{
    "entity": "Skills",
    "start": 1207,
    "end": 1218,
    "text": "& Photoshop"
},
{
    "entity": "Skills",
    "start": 1221,
    "end": 1251,
    "text": "\u25cb Web Development (HTML & CSS)"
},
{
```

```
        "entity": "Skills",
        "start": 1275,
        "end": 1284,
        "text": "Frontend:"
    },
    {
        "entity": "Skills",
        "start": 1287,
        "end": 1323,
        "text": "ReactJS, Gatsby, jQuery, JavaScript,"
    },
    {
        "entity": "Skills",
        "start": 1326,
        "end": 1360,
        "text": "HTML, CSS, Materialize, Bootstrap,"
    },
    {
        "entity": "Skills",
        "start": 1363,
        "end": 1371,
        "text": "Backend:"
    },
    {
        "entity": "Skills",
        "start": 1373,
        "end": 1378,
        "text": "Flask"
    },
    {
        "entity": "Skills",
        "start": 1381,
        "end": 1400,
        "text": "Data Visualization:"
    },
    {
        "entity": "Skills",
        "start": 1402,
        "end": 1416,
        "text": "D3, Matplotlib"
    },
    {
        "entity": "Skills",
        "start": 1419,
        "end": 1431,
        "text": "Cloud Suite:"
    },
    {
        "entity": "Skills",
        "start": 1433,
        "end": 1454,
        "text": "Google Cloud Platform"
    },
    {
        "entity": "Skills",
        "start": 1457,
        "end": 1479,
        "text": "Programming Languages:"
```

```
    },
    {
        "entity": "Skills",
        "start": 1481,
        "end": 1488,
        "text": "C++, C,"
    },
    {
        "entity": "Skills",
        "start": 1491,
        "end": 1497,
        "text": "Python"
    },
    {
        "entity": "Skills",
        "start": 1500,
        "end": 1509,
        "text": "Database:"
    },
    {
        "entity": "Skills",
        "start": 1511,
        "end": 1521,
        "text": "Oracle SQL"
    },
    {
        "entity": "Skills",
        "start": 1524,
        "end": 1557,
        "text": "Data Structures and Algorithms in"
    },
    {
        "entity": "Skills",
        "start": 1560,
        "end": 1563,
        "text": "C++"
    },
    {
        "entity": "Skills",
        "start": 1566,
        "end": 1576,
        "text": "approaches"
    },
    {
        "entity": "Location",
        "start": 1625,
        "end": 1630,
        "text": "Noida"
    }
]
```