



TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Fakultät Informatik

Transformer Technologies for analysing applicant data

Bachelorarbeit im Studiengang Wirtschaftsinformatik

vorgelegt von

Raphael Swidnicki

Matrikelnummer 298 5635

Erstgutachter: Prof. Dr. Tobias Bocklet

Zweitgutachter: Prof. Dr. Jens Albrecht

© 2021

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

Prüfungsrechtliche Erklärung der/des Studierenden

Angaben des bzw. der Studierenden:

Name: Swidnicki

Vorname: Raphael

Matrikel-Nr.: 2985635

Fakultät: Informatik

Studiengang: Wirtschaftsinformatik

Semester: Sommersemester


2021

Titel der Abschlussarbeit:

Transformer Technologien zur Analyse von Bewerberdaten

Transformer Technologies for analysing applicant data

Ich versichere, dass ich die Arbeit selbständig verfasst, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Nürnberg 19.09.2021 

Ort, Datum, Unterschrift Studierende/Studierender

Erklärung zur Veröffentlichung der vorstehend bezeichneten Abschlussarbeit

Die Entscheidung über die vollständige oder auszugsweise Veröffentlichung der Abschlussarbeit liegt grundsätzlich erst einmal allein in der Zuständigkeit der/des studentischen Verfasserin/Verfassers. Nach dem Urheberrechtsgesetz (UrhG) erwirbt die Verfasserin/der Verfasser einer Abschlussarbeit mit Anfertigung ihrer/seiner Arbeit das alleinige Urheberrecht und grundsätzlich auch die hieraus resultierenden Nutzungsrechte wie z.B. Erstveröffentlichung (§ 12 UrhG), Verbreitung (§ 17 UrhG), Vervielfältigung (§ 16 UrhG), Online-Nutzung usw., also alle Rechte, die die nicht-kommerzielle oder kommerzielle Verwertung betreffen.


Die Hochschule und deren Beschäftigte werden Abschlussarbeiten oder Teile davon nicht ohne Zustimmung der/des studentischen Verfasserin/Verfassers veröffentlichen, insbesondere nicht öffentlich zugänglich in die Bibliothek der Hochschule einstellen.

Hiermit ☒ genehmige ich, wenn und soweit keine entgegenstehenden Vereinbarungen mit Dritten getroffen worden sind,
☐ genehmige ich nicht,

dass die oben genannte Abschlussarbeit durch die Technische Hochschule Nürnberg Georg Simon Ohm, ggf. nach Ablauf einer mittels eines auf der Abschlussarbeit aufgebrachten Sperrvermerks kenntlich gemachten Sperrfrist

von 0 Jahren (0 - 5 Jahren ab Datum der Abgabe der Arbeit),

der Öffentlichkeit zugänglich gemacht wird. Im Falle der Genehmigung erfolgt diese unwiderruflich; hierzu wird der Abschlussarbeit ein Exemplar im digitalisierten PDF-Format auf einem Datenträger beigelegt. Bestimmungen der jeweils geltenden Studien- und Prüfungsordnung über Art und Umfang der im Rahmen der Arbeit abzugebenden Exemplare und Materialien werden hierdurch nicht berührt.

Nürnberg 19.09.2021 

Ort, Datum, Unterschrift Studierende/Studierender

Kurzdarstellung

Mit der stetigen Weiterentwicklung von maschinellen Lernverfahren, erschließt sich für Unternehmen ein größer werdender Einsatzbereich von Künstlicher Intelligenz. Besonders Großunternehmen haben ein steigendes Interesse an den fortschrittlichsten Informationstechnologien in diesem Bereich, um die Effizienz und im besten Fall Effektivität Ihrer Prozesse zu erhöhen. Ein Prozess der von einer steigenden Anzahl von Großunternehmen, mittels der Unterstützung von Künstlicher Intelligenz bewältigt wird, ist der der Personalauswahl.

Die Schaffung einer Künstlichen Intelligenz, die den Prozess der Personalauswahl unterstützt, ist mit Entwicklungsaufwand verbunden und lediglich im unternehmensinternem Umfeld zugänglich. Somit existieren in der Öffentlichkeit nur vereinzelt Projekte, die allgemein gehalten und auf kein spezifischen Unternehmen zu geschnitten sind.

Der Fokus dieser Arbeit liegt auf der Entwicklung einer Künstlichen Intelligenz mittels *BERT (Bidirectional Encoder Representations from Transformers)* der modernsten, maschinellen Lerntechnik in der Verarbeitung natürlicher Sprache. Diese soll dem Unternehmen *Dr. Schneider Unternehmensgruppe* eine Zeitersparnis bringen im Prozess der Personalauswahl, durch die Vorselektion einzelner Bewerber, in jeweils eine der Kategorien: "Zusage" oder "Absage" bzw. durch den Vergleich der Textähnlichkeit zwischen Lebensläufen zu einer spezifischen Stellenausschreibung. Hierfür werden in dieser Arbeit vier Variante entwickelt, evaluiert und miteinander verglichen, um schließlich die best geeignete zu ermitteln.

Abstract

With the continuous further development of machine learning methods, a growing area of application of artificial intelligence is opening up for companies. Especially large enterprises have an increasing interest in the most advanced information technologies in this area to increase the efficiency and in the best case effectiveness of their processes. One process that is being mastered by an increasing number of large companies, with the support of Artificial Intelligence, is that of personnel selection.

The creation of artificial intelligence that supports the process of personnel selection is associated with development effort and is only accessible in the internal corporate environment. Thus, there are only isolated projects in the public domain, which are general and do not suit to any specific company.

The focus of this work is on the development of an artificial intelligence using *BERT (Bidirectional Encoder Representations from Transformers)* the state-of-the-art machine learning

technique in natural language processing. This is to bring the company *Dr. Schneider Unternehmensgruppe* a time saving in the process of personnel selection, by the preselection of individual applicants, in each case into one of the categories: "acceptance" or "rejection" or by comparing the text similarity between resumes to a specific job posting. For this purpose, four variants are developed, evaluated and compared in this thesis in order to finally determine the most suitable one.

Contents

1	Introduction	1
1.1	Related Work	2
1.2	Purpose	2
2	Data	4
2.1	Data characteristics	4
2.2	Data preparation	6
2.2.1	PDF to TXT converter	6
2.2.2	Alienation of the resumes	6
2.2.3	TXT to CSV converter	6
2.3	Preprocessing for BERT	7
2.3.1	Cleaning with Regular Expression	7
2.3.2	Bert Tokenizer	8
2.3.3	PyTorch Dataloader	9
2.4	Breakdown of the data in relation to the model variants	10
2.5	Summary	12
3	Method	13
3.1	Bidirectional Encoder Representations from Transformers	13
3.1.1	BERT architecture and functionality	14
3.1.2	Transfer learning	18
3.2	Measurement of the performance of the models	19
3.2.1	Hold-out	19
3.2.2	Leave-one-out	20
3.3	Classification of resumes of different job profiles	21
3.3.1	Bert model further development for binary classification tasks (BERT v1)	21
3.3.2	Extension of BERT v1 by additional submodels and final score (BERT v2)	22
3.3.3	Resume classification only based on three submodels and a final score (BERT v3)	25
3.4	Classification for a specific job profile based on sentence similarity (BERT v4)	25
3.5	Summary	27

4 Experiments	28
4.1 Results of BERT v1-v3 experiments	28
4.1.1 BERT v1	30
4.1.2 BERT v2	34
4.1.3 BERT v3	40
4.1.4 BERT v1-v3 best results	46
4.2 Results of BERT v4 experiments	46
4.2.1 Quality planner	47
4.2.2 Production manager injection molding	48
5 Discussion	50
5.1 BERT v1	50
5.2 BERT v2	51
5.2.1 Submodel school	51
5.2.2 Submodel profession	52
5.2.3 Submodel skills	52
5.3 BERT v3	53
5.3.1 Sub-model school	53
5.3.2 Sub-model profession	54
5.3.3 Submodel skills	54
5.4 Comparison of BERT v1, v2 and v3 results	55
5.5 BERT v4	56
6 Conclusion	57
List of Figures	63
List of Tables	65
List of Listings	67
Bibliography	68

Chapter 1

Introduction

An increasing number of companies are globally in the middle of an artificial intelligence and automation transformation. As it turned out in the survey McKinsey Automation Survey 2018: "57 percent of 1,300 institutions have already started on this journey, with another 18 percent planning to kick off something within the next year" [1]. As the amount of complexity of techniques is growing more use cases appearing for various business areas. The new technology field offers companies real benefits to increase efficiency and efficacy. The consequence is that the usage of artificial intelligence gives an advantage in competing with other market participants.

Two years later in 2020, McKinsey published new edition of the survey that shows the percentage of 57 has grown up to 66 percent [2]. The findings of the newer survey also reveals that prioritizing processes together with use of artificial learning has become even more necessary to enable success. The amount of data inside companies is going up day by day what provides AI computer systems more training material to achieve best results in duplicating human behaviour.

Another insight is that successful organizations proceed to classify employees as much relevant as technology. It's not about replacing humans in every field by artificial intelligence it's about work alongside and relieve employees, so that they can focus on more complex tasks which can not be solved by computer systems.

Human resource represents one core area of every company, due to employees create additional value. Companies want to scale their business or suffer turnover of workforce based on the war of talents [3, p. 48]. Thus, it is mandatory to pay special attention to one of the main tasks: the recruiting. Depending on how the work sharing looks like the human resource managers have several other tasks. With could be strategic personnel planning, controlling and implementing labor law requirements or active sourcing. To decrease or in the better case to avoid a work overload, companies apply computer-aided systems to automate the selection procedure of applicants as much as possible.

The common use case is to pass through a preselection process which is applied to the vast amounts of applications and done by the computer. With this an first impression is

created and the recruiter can focus on the most likely applicants. So far, the solutions usually have based on predefined keywords which were compared with the application text of the candidate. Meanwhile there is instead a change to use artificial intelligence to deliver solutions.

1.1 Related Work

The research work for the field of developing programs for pre-selection of applications based on computer systems are often a part of internal company processes. Nonetheless, one public paper has similarities with this work and will be introduced in detailed with this subchapter.

The name of the paper is "End-to-End Resume Parsing and Finding Candidates for a Job Description using BERT" [4] which was published in October 2019. The research team presented an end-to-end solution for automating ranking candidates based on their suitability to a job description. For this task the researchers mentioned two main stages:

The first stage is a resume parser which is capable of extracting all the information of a PDF resume file to an readable CSV file format for BERT. Often: "resumes come in myriad formats, and simply parsing the resume correctly is a very difficult task for a machine" [4]. One solution to tackle this problem was to built heuristic rules and a classifier trained on LinkedIn data, which follows standard format structure. The purpose of the classifier was to identify the sub-categories of resumes of any format structure. After the step of classifying all the segments of text were converted from a non-LinkedIn resume to a LinkedIn format resume. The team managed to gather data by extracting information from an assortment of 715 LinkedIn-style and 1000 non-LinkedIn-style resumes.

For the second stage another BERT classifier was trained by simulating the job description that a company would be looking to hire individuals for. Positive samples were created by using the candidate's experiences at a previous role. The experiences between different candidates were taken as negative samples [5, p. 4]. The model predicted whether the two job descriptions were of the same candidate or not. Finally, with this method the results achieved 72.77% accuracy in predicting whether two job descriptions belong to same person or not. The idea behind this approach was to create a model which can be used to compare similarity between the job descriptions of a candidate and the company.

1.2 Purpose

In this research work, different models based on BERT will developed and tested in order to finally find the most suitable one for the pre-selection process of applicants of the company

Dr. Schneider Unternehmensgruppe. Thereby the provided data of resumes will be converted and preprocessed to ensure the highest data quality level for generalization. Among other things, a parser will also be developed for this purpose. The parser will extract either all resumes which are labeled as acceptance or rejection from there PDF files to each a single row of a csv document.

In total, four varieties will be elaborated. The first three aim at developing models which can evaluate new resumes by means of the labeled resumes in "acception" or "rejection". Furthermore, for the second and third variant, the use of sub-systems, which consist of trained models of one of the three categories: school history, professional history, skills, is investigated. Finally, the predictions of the individual sub-systems should provide a final result about the suitability of the applicant by using different score functions. BERT v4 variant will compare the text similarity of *one specific job description* and the resumes which refer to this.

Chapter 2

Data

The better the data the more successful the prediction is. Data preparation and cleaning represent a critical process to get high-quality training data. This chapter reveals characteristics of the data, the steps to change the format and make it understandable for the BERT Tokenizer. Furthermore, it shows up the mandatory alienation of the raw data and discusses the different data requirements of the individual BERT variants in terms of quantity and type. For a better understanding of the different data requirements of the variants, figures are shown at the end of this chapter.

2.1 Data characteristics

The used raw data in PDF file format (table 2.1) were provided by the Dr. Schneider company in a directory within the in-house network. In total: 83 labeled data of applicants, 42 rejections (of which three were in English) and 41 acceptance (of which two were in English). The development of the models is based on the approach of a German-language BERT model (Huggingface: bert-base-german-cased), for this reason the five English-language applicant data were not considered further.

	application	resume only	sum
rejections	9	33	42
acceptance	5	36	41
german-speech data in total	14	64	78
different job profiles	-	-	23

Table 2.1: Raw data segmentation

Another significant feature of the data is the diversity of different job profiles. In total, the datasets listed in table 2.1 include 23 job profiles. Strictly speaking, one type of the job descriptions are unsolicited application so they are not assignable to one specific job

description. A closer examination of the data leads to the assumption that due to the high variance of job profiles in relation to the small amount of data, generalization error could occur. The challenge is to develop a model that is able to classify applicants as potential company employees beyond the job requirement.

The textual content of a resume can be divided into four categories as follows:

1. personal data: this includes the name, place of residence, contact information, etc.
2. school history: the history of the schools attended.
3. professional history: the history of professional activities performed.
4. skills: this includes various skills such as computer skills in using a particular software.

This division of the resume data into the above categories will be essential for the development of BERT variants 2 and 3 in the later course of the thesis.

The fourth variant (BERT v4) was based on the approach to take data of specific job profiles to find the resumes that got the highest similarity to a given job advertisement. Two job profiles were selected to further explore this approach. 2.2 shows the job profiles and the related amount of used data divided into the two categories: rejections and acceptance.

job profile	Fertigungsleiter Spritzguss	Qualitätsplaner
rejections	5	9
acceptance	5	3
job advertisement	1	1

Table 2.2: Used resume data for BERT v4

Note: The data sets for the job profile: "Fertigungsleiter Spritzguss" were made available towards the end of the thesis in order to be able to make a better assessment of this approach. This dataset was used only for the development of BERT v4 and was not included in Table 2.1.

The raw data contained sensible personal data (see bullet point 1. personal data), because of this an alienation process was mandatory before the data were approved to leave the borders of the in-house network. All the personal data could be first removed after they were parsed with a self-coded parser program (section 2.2.1).

2.2 Data preparation

2.2.1 PDF to TXT converter

With the help of the written program PDF to TXT compiler the resumes could be converted into an editable text format. The converter function requests the path for each PDF file. Optionally the converter function has the input parameters:

- **format:** this parameter allows to convert the content of the pdf documents either to text format or to HTML format.
- **codec:** the value set for this parameter defines the codec to be used. For the purposes of this dataset, Unicode encoding was used as it is the most widely used and therefore the error rate for unrecognized characters is the lowest.
- **password:** in case a document needs a password this parameter would be used to open it. This was not the case for the resumes of this thesis.

To pass every pdf file of the given directory an iteration was implemented with the condition to run as long as the iteration has not passed the number of resumes. After the compiler has finished all the new txt files with the converted resumes are locatable in the sub directory /txt.

2.2.2 Alienation of the resumes

As already mentioned in chapter 2.1 the dataset contains personal information about the applicants which first have to be removed before any further development steps could follow. A possible approach for the removal of personal data was to develop a program that removes the associated values from the text documents based on keywords such as name, place of residence, street. However, these keywords varied as well as the position of the associated values. Furthermore, resumes also contain personal data without identifying keywords for the values. Thus, it turned out to be very difficult to perform this task automatically with the help of a program. The most effective solution for the alienation process was pragmatic nature. Thanks to the small amount of data the resumes could be revised manually.

2.2.3 TXT to CSV converter

For the development of BERT models, all individual resumes that existed in .txt and belonged together were written in each case in one .csv file. So that every resume can be used as a sentence, due to BERT is a sentence model and can only be trained with sentences. In this sense, a second converter was implemented, which transfers and saves all .txt files

contained in the path of a specified directory into a common .csv document by means of an iteration loop. The converter function expect as input: the resumes (also called sentences due to the fact that BERT is trained by using tokenized sentences as input) and the name of the column heading of the .csv file which includes all the resumes or sentences. Only after this conversion step was completed the data could be processed further to serve then as data for the feature embedding which will be described in chapter 3.1.1.1.

2.3 Preprocessing for BERT

2.3.1 Cleaning with Regular Expression

Cleaning data means the removal or fixing of missing data. In the previous subchapter about the alienation, the first data has been cleaned in the sense of anonymization. Another kind of data cleaning is the removal of unnecessary characters. The Python standard library provides for this case the regular expression module. A regular expression define a set of strings that matches it. The functions in the module review if a specific string matches a given regular expression and for example remove this substring to replace it with another one (like an empty string to remove unwanted string parts). [5]

```

1 def text_preprocessing(text):
2     # Remove whitespace after question mark
3     re.sub(r'([te\?])\s+', r'\1', text)
4
5     # Remove all whitespaces between words
6     text= re.sub(r'[:w\s?:w\s]*', '', text).strip()
7
8     # Remove characters which appears more than once in a row
9     text= re.sub(r'(\-)\1*', ' ', text).strip()
10
11    # Remove question mark
12    text = re.sub(r'\?', ' ', text).strip()
13
14    return text

```

Listing 2.3.1: Cleaning with Regular Expression

Listing 2.3.1 illustrates the text_preprocessing function which is a part of the overall preprocessing_for_bert function that will be explained in chapter 2.3.2 in detail. One case which frequently showed up was that whitespaces appears between the characters of a single word for example: "S C H U L A U S B I L D U N G". This misrepresentation could

be corrected with the regular expression operation in line 6 which return the word to its original form. Single question mark, white spaces behind them and other character which appears one after the other multiple were removed before the tokenizing.

2.3.2 Bert Tokenizer

The transformers library includes the BertTokenizer which enables the usage of tokenizer function for BERT. Before the converted and alienated data of resumes could be used to build the different models it has to be tokenized. For this purpose an instance of the class BertTokenizer have to be initializes. Every sentence or specifically data row which includes the information of a resume runs through the iteration in line 7 (Listing 2.3.2 which execute the *tokenizer.encode_plus* function of the previous created object [6]. This tokenizer function returns a dictionary of values. In this case the needed output of vales were the *input_ids* and *attention_mask*. The more detailed meaning is explained for these values of the BERT model in chapter 3.

```

1 from transformers import BertTokenizer
2 tokenizer = BertTokenizer.from_pretrained
3     (path_model, local_files_only=True)
4
5 def preprocessing_for_bert(data):
6     ...
7 for sent in data:
8     encoded_sent = tokenizer.encode_plus(
9         text=text_preprocessing(sent),
10        add_special_tokens=True,
11        max_length=MAX_LEN,
12        pad_to_max_length=True,
13        return_attention_mask=True
14    )
15    ....
16 return input_ids, attention_masks

```

Listing 2.3.2: Preprocessing data of resumes

Furthermore listing 2.3.2 points out in which way the data were transformed before saving the values in the dictionary of the BertTokenizer object. Each resume text were preprocessed by *text_processing* function that was mentioned in chapter 2.3.1.

The add special tokens parameter add a start [SEP] and end [CLS] token for every resume or row of the .csv data. In addition the allowed amount of tokens for each resume was set

with the parameter `max_length` in line 11. For a better understanding of which length is useful, the number of words for all resumes is displayed in a graph 2.1.

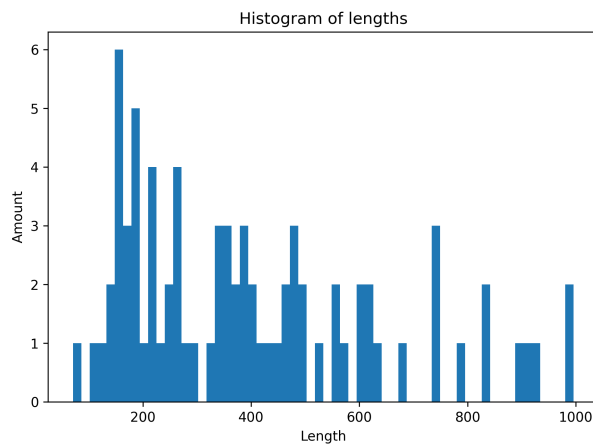


Figure 2.1: Word lengths of the individual resumes

It can be seen that most resumes are between 0 - 600 words in length, so the maximum allowed value of 512 was used for the `max_length` parameter.

The `encode_plus` function also includes the padding for every resume to the number of tokens that `MAX_LEN` allows. The padding avoided the loss of spatial dimensions [7]. This means that through filling the shorter resumes up with tokens that represents the value 0, the models were capable to detect features even in the edges of the input data. In the end the `preprocessing_for_bert` function returns tensors of all generated `input_ids` and `attention_masks`.

The function was used to tokenize the test and validation data and convert them into tensors.

```

1 train_inputs , train_masks = preprocessing_for_bert(X_train)
2 val_inputs , val_masks = preprocessing_for_bert(X_val)

```

Listing 2.3.3: Preprocessing test and validation data

2.3.3 PyTorch Dataloader

The PyTorch Dataloader brings a solution for parallelizing the data loading process with automatic batching. With the help of the Pytorch Dataloader data loading process boosts up in speed. With the usage of the batch size which is an important hyperparameter of

the PyTorch Dataloader it was possible to increase significantly the learning success of the neuronal network [8].

Every approach of creating models to classify resumes in this thesis were realized with batches and tensors. A PyTorch Tensor is almost the same as a numpy array. It also represents a generic n-dimensional array but the biggest difference of the PyTorch Tensor is that it also can run on GPU. Furthermore the data loader gets assigned tensors which means first they have to be initialized before batching can be used at all.

Listing 6 gives an overview of the source code which was responsible for:

- creating tensors out of the train and validation labels (line 5-6).
- defining the batch size (line 9).
- initialization of a tensor dataset object for the train as well as for the validation data (line 12 and 18). With the input parameters from the invoke of the pre-processing function with the test and validation data.
- determination whether the data should be shuffled or not through creating a random sampler and sequential sampler object (line 13 and 19).
- initialization of a data loader object with the TensorDataset train data/validation data as input parameter, the sampler and the defined batch size.

2.4 Breakdown of the data in relation to the model variants

Finally, figure 2.2 shows an overview with the assignment of the data used for the respective developed BERT variants of this thesis. A more detailed explanation of the approaches is given in chapter 3.3. This section is primarily intended to provide a better understanding of the relationships at the data level.

For the first variant (BERT v1) the input consists of two .csv files:

One is the rejections.csv file, which contains all resumes that have been labeled as rejections. The other one is the acceptance.csv file, which contains all resumes that have been labeled as acceptances. This is the data basis for the creation of Model 1.

For the second variant (BERT v2) the input also consists of the same input data like for the previous mentioned BERT v1 variant but furthermore 6 additional .csv files were used, which consist of the categories mentioned in chapter 2.1. Two files are assigned to each category: one with the rejected and one with the accepted resumes.

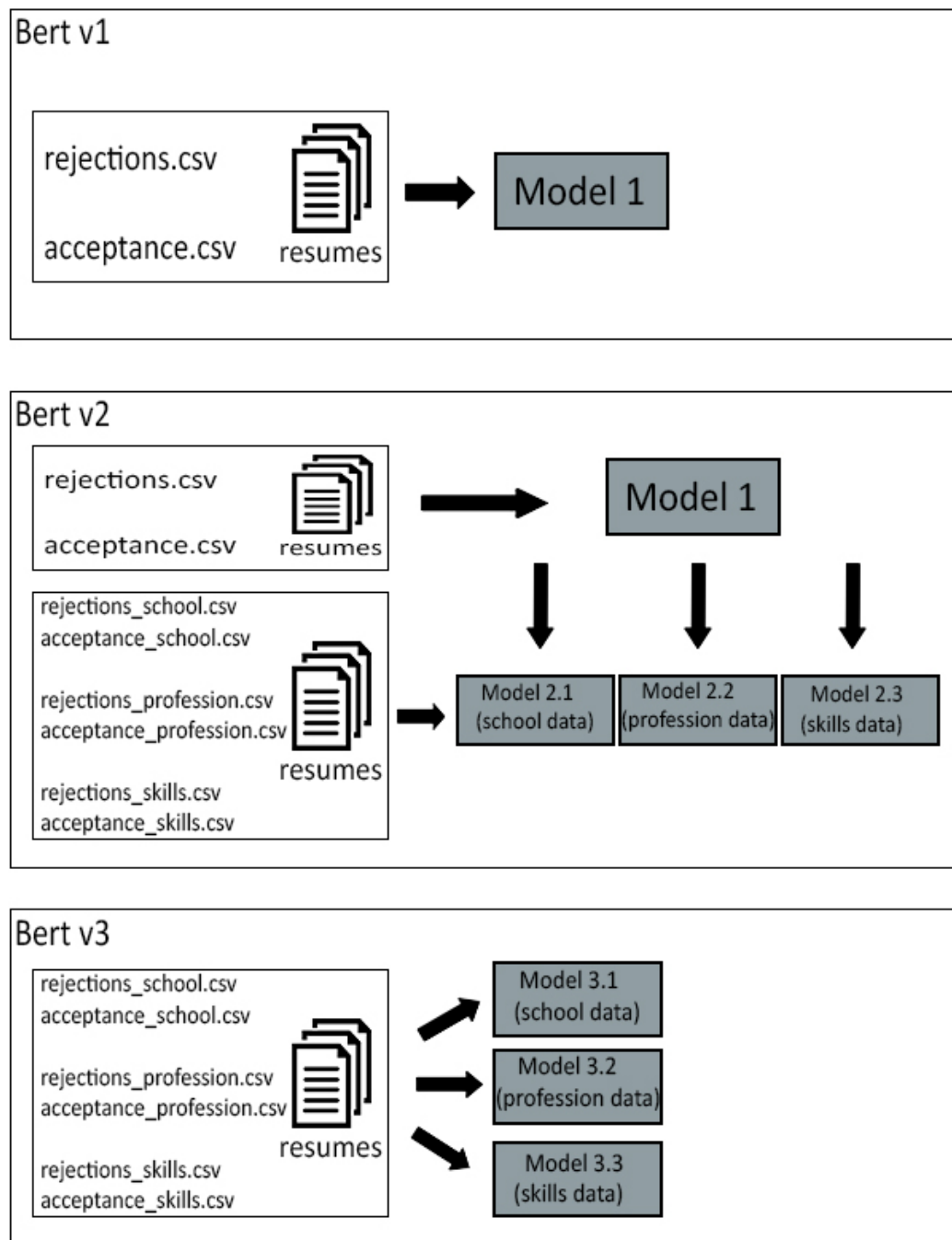


Figure 2.2: Breakdown of data for the variants 1-3

The input of third variant (BERT v3) was only represented by the 6 .csv files which were mentioned with regard to BERT v2.

The presentation of the data and their intended use look fundamentally different for the fourth variant. In contrast to the previous variants, the job descriptions mentioned above were also used to generate a vector for each (see figure 2.3) *vector_jd*. At the same time, the associated resume data was used to generate a vector for each resume submitted for the job description.

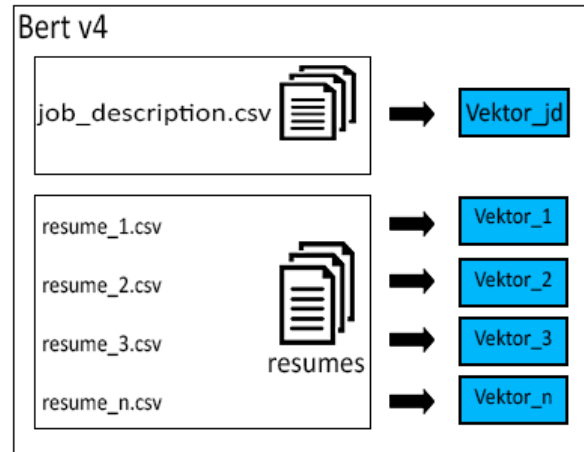


Figure 2.3: data level BERT v4

2.5 Summary

This chapter was a dive into the various data and their characteristics. It provides an understanding of the binary classification problem and how the data is separated for the underlying submodels. It also introduced the two converters and explained how they help put the data into the format needed for BERT. This includes the presentation of the pre-processing function and the associated tokenization. Furthermore, the Pytorch Dataloader and the concept of Tensors were presented, as they play a relevant role to speed up the data loading process and to be able to influence the performance of the model via batch sizes.

Finally, a graphical overview of the data used and the associated models or vectors of the individual BERT variants was created. This helps to better understand more complex issues later in the course of the thesis.

Chapter 3

Method

In the following chapter, the structure and the functionalities of the language representation model BERT are explained. Based on the use of BERT, different methods were applied to train neural networks for the classification of the applicant data. These different methods, which use the data preprocessed in chapter 2, are also shown individually. Furthermore, the two cross-validation methods: hold-out and leave-one-out cross-validation are explained, which were used in the thesis to obtain reliable and unbiased results for the individual model performance.

Machine learning models use numerical vectors or matrices to generate related output vectors to that input. Before the prepared data can be used, features have to be created. In the following chapter, the thesis will also describe in more detail how feature embedding was implemented in order to develop the various classification models with the applicant data and BERT.

After dealing with this chapter, the basis is created to present the results of the development approaches and the related experiments.

3.1 Bidirectional Encoder Representations from Transformers

The language representation model BERT was published by the research scientists of Google in November 2018:

"BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results [9]."

3.1.1 BERT architecture and functionality

The following section provides an overview of BERT's architecture. After the high-level look, the individual components will be described more in detail and also in directly relation to this thesis.

The first part of the architecture represents the input embedding and positional encoding area (figure 3.1) which have to be proceeded before the more complex encoders taking place. The transformer encoder layers have multiple heads, for example fully-connected neural networks augmented with a self-attention mechanism. For each tokenized word of a resume, three values are calculated: a key (K), a value (V) and a query vector (Q). At the end all these values are merged in the same layer and then passed through a fully-connected layer [10]. More will be explained in detail in the encoder subchapter.

The transformer itself was introduced in the paper "Attention Is All You Need" in December 2017. The whole transformer architecture based on an encoder and decoder. However, BERT only uses a encoder for this reason the focus of the explanation is on the encoder.

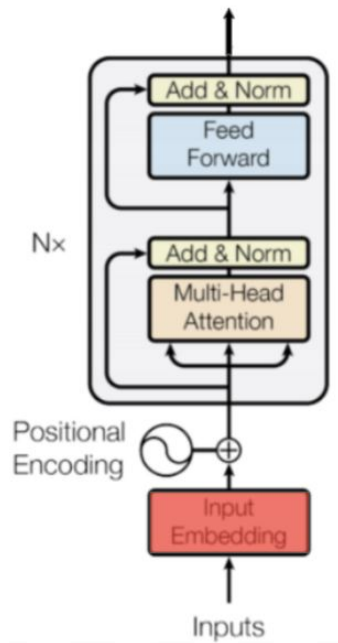


Figure 3.1: BERT architecture [11, p. 3]

3.1.1.1 Feature embedding

From the perspective of the information flow all starts with the inputs. In the specific case of this thesis the inputs are the resume data which were shown in more detail in chapter

2. To train BERT the resumes had to be first tokenized with the `preprocessing_for_bert` function (chapter 2.3.2).

Tokenizing means to decompose the text of a resume into its individual words and to map each of these words to a unique integer in the corpus vocabulary:

["erfahrung", "Software", "und", "Testingenieur", "bei", "Bosch"] \rightarrow [34, 90, 15, 684, 55, 193]

The integer value of each mapped word also belongs to an embedding dimensional vector like it is pictured on figure 3.2. The vectors are necessary for a BERT model to learn during the training process.

$$\begin{aligned}
 34 &\rightarrow E[34] = [123.4, 0.32, \dots, 94, 32] \\
 90 &\rightarrow E[90] = [83, 34, \dots, 77, 19] \\
 15 &\rightarrow E[15] = [0.2, 50, \dots, 33, 30] \\
 684 &\rightarrow E[684] = [289, 432.98, \dots, 150, 92] \\
 55 &\rightarrow E[55] = [80, 46, \dots, 23, 32] \\
 193 &\rightarrow E[193] = [41, 21, \dots, 74, 33]
 \end{aligned}$$

Figure 3.2: Corresponding vectors to tokens

After the corresponding vectors are available each of the vectors are putted together to get a matrix Z of dimensions.

$$\begin{array}{l}
 \text{Erfahrung} \\
 \text{Software} \\
 \text{und} \\
 \text{Testingenieur} \\
 \text{bei} \\
 \text{Bosch}
 \end{array}
 \begin{pmatrix}
 < & - & d_{emb_dim} & - & > \\
 123.4 & 0.32 & \dots & 94 & 32 \\
 83 & 34 & \dots & 77 & 19 \\
 0.2 & 50 & \dots & 33 & 30 \\
 289 & 432.98 & \dots & 150 & 92 \\
 80 & 46 & \dots & 23 & 32 \\
 41 & 21 & \dots & 74 & 33
 \end{pmatrix}$$

Figure 3.3: Matrix Z of dimensions

This matrix includes the input length x embedding dim. The max. value for the input length could be 512. In case that a sequence which contain the data of a resume does not consist of the equity amount of words the remaining tokens slots are filled up with "<pad>" tokens. This was secured through setting the parameter `pad_to_max_length = TRUE` like it was illustrated and explained why in chapter 2.3.2 of the thesis.

In the last step before the encoder will be introduced in detail the positional encoding must be mentioned (as it can also be seen in figure 3.1).

$$p_{i,j} = \begin{cases} \sin\left(\frac{i}{10000^{\frac{j}{d_{emb_dim}}}}\right) & \text{if } j \text{ is even} \\ \cos\left(\frac{i}{10000^{\frac{j-1}{d_{emb_dim}}}}\right) & \text{if } j \text{ is odd} \end{cases}$$

The variable i is used for the token position in the sequence and j for the embedding feature position. The decision about choosing the sinusoidal or cosinusoidal depends on the value of the variable j . If the value of the variable is even the sinusoidal function takes place otherwise the cosinusoidal one [11, p. 5].

So, additional to increase the learn success sinusoidal and cosinusoidal functions are used which allow positions to be represented as linear combinations of each other. This enables the network to learn relative relationships between the token positions.

$$\begin{array}{l} \text{Erfahrung} \\ \text{Software} \\ \text{und} \\ \text{Testingenieur} \\ \text{bei} \\ \text{Bosch} \end{array} \begin{pmatrix} \sin\left(\frac{0}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \cos\left(\frac{0}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \sin\left(\frac{0}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \cos\left(\frac{0}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \dots \\ \sin\left(\frac{1}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \cos\left(\frac{1}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \sin\left(\frac{1}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \cos\left(\frac{1}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \dots \\ \sin\left(\frac{2}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \cos\left(\frac{2}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \sin\left(\frac{2}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \cos\left(\frac{2}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \dots \\ \sin\left(\frac{3}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \cos\left(\frac{3}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \sin\left(\frac{3}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \cos\left(\frac{3}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \dots \\ \sin\left(\frac{4}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \cos\left(\frac{4}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \sin\left(\frac{4}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \cos\left(\frac{4}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \dots \\ \sin\left(\frac{5}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \cos\left(\frac{5}{10000^{\frac{0}{d_{emb_dim}}}}\right) & \sin\left(\frac{5}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \cos\left(\frac{5}{10000^{\frac{2}{d_{emb_dim}}}}\right) & \dots \end{pmatrix}$$

Figure 3.4: Previous Matrix Z together with functions to add positions

The final matrix can be described with the equation:

$$X = Z + P$$

3.1.1.2 Encoder

In this thesis the BERT base model was used with 12 Encoder blocks. The task of the encoder is to identify relationships in the input representations and encode them to generate an output. All the encoder are chained and increasing the capability of the neural network to understand complex coherences of words of the input.

The Multi-Head Attention is the first part of the encoder which is in charge to compute attention different times and bring them together. Attention is a technique that should emulate the cognitive attention of a human. The effect focus more on the important pieces of the input and ignores the rest. Furthermore this enables to spend higher amount of computing power on more important data.

The computations of the attentions occurred parallel like it is illustrated in figure 3.5. In the illustration V, K, Q which have already been mention in the beginning of this chapter stands for different weight matrices. The three values V, K and Q then are used to compute the Scaled Dot-Product Attention [11, p. 3]:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

For a better understanding the next lines will show an example on the first word of a resume (sentence):

Professional history 2008 - 2011 Bachelor of Science University Berlin

Query (Q) stands for the input word vector for token "Professional" which is the first one in current example sentence.

The keys (K) are the input word vectors for the other tokens (also for the query token).

history;2008;-;2011;Bachelor;of;Science;University;Berlin + Professional

(The semicolon is used for this example to separate the input word vectors.)

The word vector of the query is then Dot-Product with the word vectors of each of the keys, to get 10 scalars alias the weights which are then scaled [12, p. 3]. The Dot-Product of the query is computed with all keys, divide each by: $\sqrt{d_k}$ The dot product is scaled by this division to prevent it from getting huge as d_k increase.

In the next steps a softmax is applied on the weights to normalize 10 weights to values between 0 and 1. The initial 10 input word vectors (which are the values (V)) are summed in a weighted average together with the normalized weights. The result is a single output word vector representation of the word "Professional". This procedure repeats to get a vector for each of the remaining 9 tokens. Finally, the result of the scaled dot-product is concatenated and applied to last linear layer to produce the multi-headed attention output.

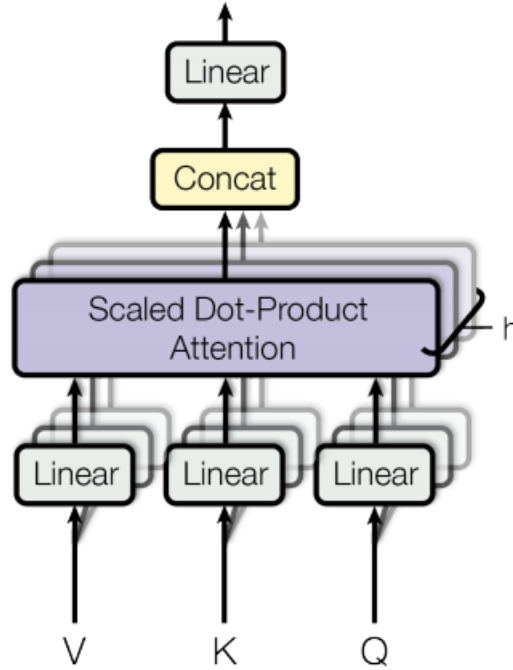


Figure 3.5: Multi-Head Attention [11, p. 4]

The second part of the encoder is a fully connected feed-forward network which got two linear transformations with a ReLU activation in between [11, p. 5].

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

The fully-connected layer uses the weights W_1 and biases b_1 and performs a ReLU function (max with zero), and then uses a second fully-connected layer with weights W_2 and biases b_2 .

The output of an encoder n is the new input for next encoder this is the case until the last of the 12 encoders are done with computation. After this the final embeddings are created. Every layer of the BERT network has a hidden size of 512×768 . 512 represents the number of tokens per each sentence and 768 is the numerical representation of a single token.

The transformer allows each token to look at all the other tokens at the same time this is an advantage to a RNN or LSTM which can only look at the tokens to the left.

3.1.2 Transfer learning

As already mentioned in chapter 2 the data amount was very small and the variety of job profiles very high. This shortage of training data does not allow to built a whole new

neuronal network with pre training. Due to this fact the language model: bert-base-cased from huggingface together with the concept of transfer learning was the solution to handle this problem. The model consists of 12 GB training Data from the latest German Wikipedia dump (6GB of raw txt files), the OpenLegalData dump (2.4 GB) and news articles (3.6 GB)s [13].

Normally, a BERT model would first have to be created from scratch using pre-training and the two associated tasks: masked language model and next sentences prediction [14, p. 4]. This requires a high data approach and computational power as well as time. Transfer learning eliminates the need for pre-training by resorting to a pre-trained model. The idea behind transfer learning is to profit of the huge trained neuronal network structure by adding an additional classifier layer as the last layer before the output.

3.2 Measurement of the performance of the models

For the optimization and evaluation of the different models in terms of their performance, two cross validation methods were used in this thesis: hold-out and a leave-one-out.

3.2.1 Hold-out

The idea of hold-out is to split up the dataset into a train and test set. The training set is used to train the model. The test set is then used for measuring how well that model performs on unseen data. Usually when using the hold-out method the train set consists of 80% of data and the remaining 20% are represented by the testing set. For the development of the different models the size of test set was increased to 25% due to the fact that the data set was very small and the goal to get a higher reliability of the hold-out procedure [15].

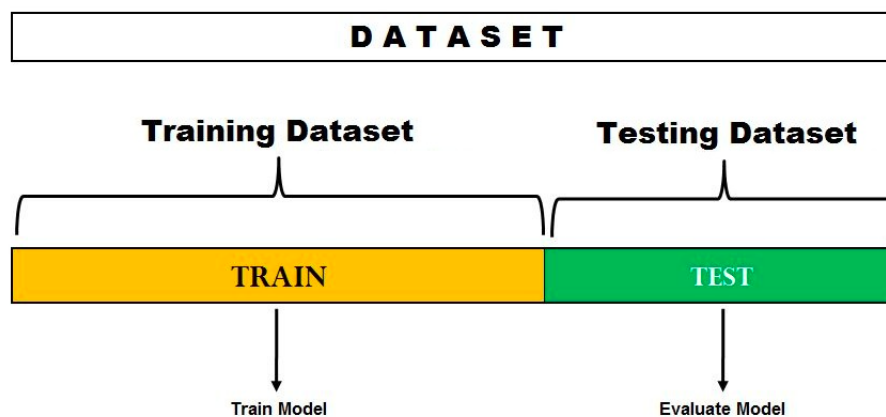


Figure 3.6: Hold-out method for model evaluation [15]

This thesis includes several experiments in the next chapter to find the best performing models. For finding those models the hyper parameters tuning process was applied by using the hold-out method. After the hyper parameters constellation for the best reachable results was identified the model was trained and saved by using the whole data set.

3.2.2 Leave-one-out

Leave-one-out is a form of cross validation. Cross validation means to split up a dataset randomly into 'k' groups. The model is trained on the training set and evaluated on the test set. There are several runs and each run has one k-fold as the test set and the other folds are used as the training set. This process is repeated until each unique k-fold has been used as the test set.

The method is popular to evaluate the performance of a classification algorithm when the number of instances in a data set or the number of instances for a class value is small [16, p. 4]. When considering the total of 78 data sets for BERT variants 1 to 3, the leave-one-out cross validation seems to be a very plausible means to determine the performance.

The number of folds is equal to the number of instances in the data set. One selected instance is used as a single-item test set while all other instances are part of the test set. After the learning algorithm has been applied once for each instance the leave-one-out cross validation finished.

At the programming level, the LeaveOneOut module of sklearn was implemented, an object was instantiated and then a loop was created.

```

1 for train_index , test_index in loo.split(X):
2     print( "TRAIN:" , train_index , "TEST:" , test_index )
3     X_train , X_test = X[train_index] , X[test_index]
4     y_train , y_test = y[train_index] , y[test_index]
```

Listing 3.2.1: iteration for leave one out cross validation

The following numbers represents the single items which where used to create a train data set and the last item stands for the sample to evaluate.

```

TRAIN: [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77] TEST: [0]
```

The first item "[0]" serves as evaluation sample for the first loop pass and with each new loop pass the numbers increase "[k+1]" until it was the turn of the last sample "[77]".

In chapter 4, which shows the results for the different approaches, the roc-curves (receiver operating characteristic curves) will be presented which have been created by using the leave-one-out method to plot the performance of the classification algorithms. With the use of the roc curve and the leave one out method, the different models can be evaluated for their generalization ability and the threshold between true positive and false positive can be determined.

3.3 Classification of resumes of different job profiles

This section addresses the various methods of creating as well as using BERT models to evaluate resumes of potential applicants for their suitability as candidates. For this purpose the previously mentioned variants BERT v1 to v3 and their individual special features are explained in more detail in under sections. On the other hand After that, BERT v4, which differs in its approach, is explained in an extra subchapter.

3.3.1 Bert model further development for binary classification tasks (BERT v1)

The first three variants based on the idea to use the bert-base-german-cased model which consists of 12 transformer layers (as mention in the transfer learning chapter). Every transformer layer receives the list of token embeddings, and produces the same amount of embeddings on the output. The output of the final transformer layer is used as the features of the sequence to feed a classifier. The class BertClassifier was implemented to enable the instantiate of the one-layer feed-forward classifier.

For the BERT v1 variant, all labeled datasets were used together to train the 13 layer network to obtain a model capable of making predictions about the suitability of a resume (illustrated in Figure 3.7).

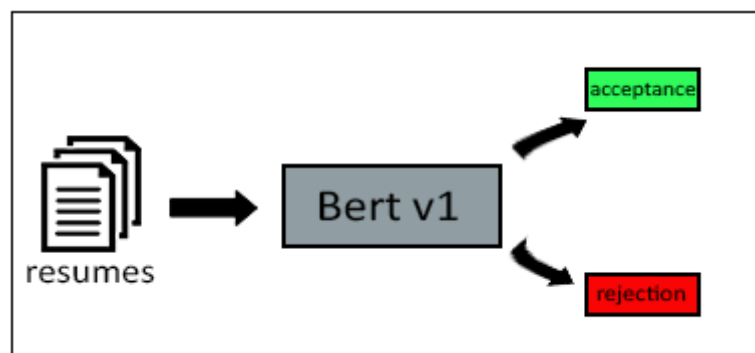


Figure 3.7: BERT variant 1

The box in the middle of the illustration represents the whole BERT model which received as input all the mentioned resumes from chapter: 2.1 data characteristics. The trained model is then used to take unseen resume data as an input parameter and generate an output at the end. The prediction for the desired records is done using the *bert_predict function*. This puts the model into evaluation mode, assigns all batches to the gpu and iterates through them to determine the vectors that correspond to the input data. For better understanding of the results, vectors of probabilities are created where the probabilities of each value are proportional to the relative scale of each value in the vector. This is done in this thesis with the help of the *softmax activation function* [17]:

$$S(y)_i = \frac{e^{y_i}}{\sum_{j=1}^n e^{y_j}}$$

Each number's value is between 0 and 1 valid value range of probabilities for the output of the softmax function. The entire output vector sums to 1.

For example the input logits y_i for softmax function could be 1.4 (predicted value for label 0 $\hat{=}$ acceptance) and 0.6 (predicted value for label 1 $\hat{=}$ rejection). After applying the function (with the above formula), the output would be 0.6 and 0.4, and the sum would be equal to a probability prediction of 100.

3.3.2 Extension of BERT v1 by additional submodels and final score (BERT v2)

After creating a model which predict the type of a resume based on the whole data of it a further approach seemed to be interesting.

By using and extending the first model variant, three submodels were created. Each time the best model of the first variant, which was saved, was used to add another classifier layer. The data used for this consisted of copies of the resumes, which were revised. Every resume contained data on only one of the following three categories:

1. school history
2. professional history
3. skills

In order to create the models with the best performance, a series of tests were carried out with various hyperparameters, which will be presented in the next chapter. The result of a series of experiments or training and evaluation runs was one model for each of the three categories. Thereby all individual models are based on the BERT v1 model as shown in Figure 3.8.

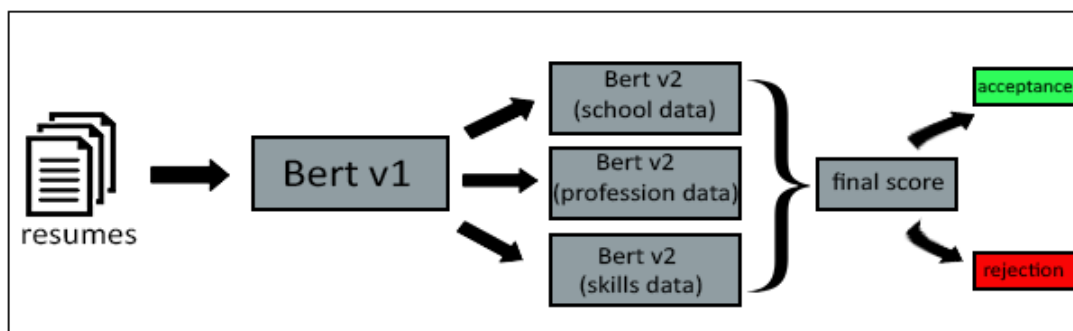


Figure 3.8: BERT variant 2

Further to be able to make predictions for a resume with the BERT v2 variant, functions were written. In contrast to the first variant, two additional functions had to be implemented. One of them is the *splitted_prediction function*, which receives as input parameter a folder path containing the .csv files with the partial data of the resumes of the three mentioned categories. Inside the function a prediction is made by invoking the *create_dataloader_and_predict* function which creates for each data set a dataloader and invokes the *bert_predict* function to get predictions for every data set. In the end three lists are returned by the *splitted_prediction function*.

These serve as input parameters for the *final_score function*, which sums up the three probability values for a classification of the resumes as rejection or acceptance.

```

1 def final_score(probs_school_achievements ,
2 probs_professional_education , probs_skills , scoretype):
3
4     if scoretype == "unbiased":
5         final_score = probs_professional_education +
6         probs_school_achievements + probs_skills
7
8     if scoretype == "heuristic":
9         final_score = (probs_professional_education * 0.6)
10        + (probs_school_achievements * 0.3) + (probs_skills * 0.1)
11
12    final_score = np.argmax(final_score , axis=1)
13
14    for idx, x in enumerate(final_score):
15        if x == 1:
16            print("Resume_", idx, "_classified_as:")
17            print("Predicted_label_type:1->rejection\n")
18            final_score = 1
19        else :
20            print("Resume_", idx, "_classified_as:")
21            print("Predicted_label_type:0->acceptance\n")
22            final_score = 0
23
24    return final_score

```

Listing 3.3.1: final score function

The function is applicable for BERT v2 and BERT v3 and has the additional input parameter: *scoretype* which allows to choose between *unbiased* and *biased* score calculation.

An unbiased calculation of the final score means every probability value of the different categories is weighted in the same way. The heuristic approach, on the other hand, allows categories to be given varying degrees of relevance to the final prediction. This makes sense because, for example, the data set of professional experience can be more relevant in the selection process of applicants than that of additional skills.

Before creating the output result for every resume the two prediction values for each label are removed and only the number of the label with the highest probability is saved.

3.3.3 Resume classification only based on three submodels and a final score (BERT v3)

The last variation is quiet similar to the previous one. But instead of building sub models for the three categories on top of the BERT v1 model all the three models were trained and evaluated without. For this purpose the 12 layers of the bert-german-cased model was extended with a classifier layer for each sub model.

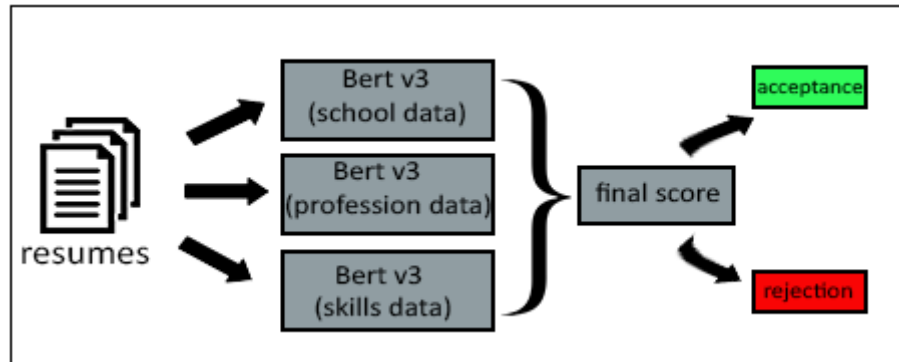


Figure 3.9: BERT variant 3

3.4 Classification for a specific job profile based on sentence similarity (BERT v4)

The approach behind the BERT v4 variant differs fundamentally from the previous ones except for the fact that the German huggingface model is also BERT-base-cased. However, as already shown in table 2.2 of chapter 2.1, additional data were used, which were subsequently made available in order to obtain a better significance. Besides, there was no direct binary classification problem for which a BERT model with labeled data had to be further trained.

To explore another way of developing a program that is able to support the application process with machine learning, the concept of text similarity was used.

In this thesis the cosine similarity was used to measure the similarity between the vector that represents a job description and the vector that is created for a resume document.

$$sim(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$$

x and y are the two vectors for comparison. $||x||$ is the euclidean norm of vector and $||y||$ stands for the euclidean norm of vector y . From a conceptually point of view both could be mention as the length of the vectors.

The result of the formula is the cosine of the angle between vectors x and y . If the result is close to 0 it means that the job description and resume vector are at 90 degrees to each other and there is no similarity. Conversely if the cosine value is close to 1 the similarity is high: the smaller the angle means a greater match between a job description and resume vector [18]. In conclusion, it was necessarily to build a function that is capable to measure the similarity for each resume vector and the corresponding job description vector and finally display the similarity in percentage for each resume vector (illustrated in figure 3.10).

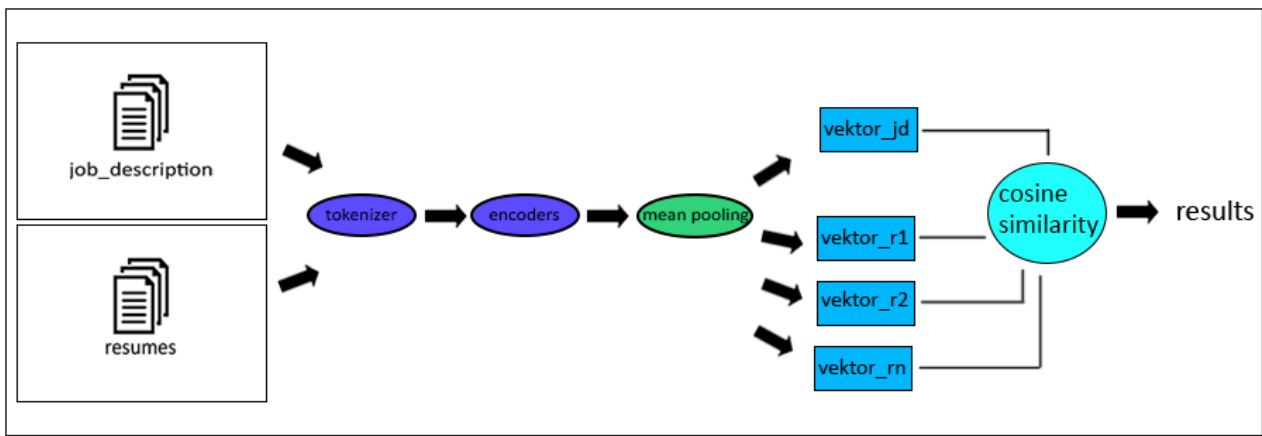


Figure 3.10: BERT variant 4

Before the cosine similarity can be determinate it is necessary to tokenize and encode the job description and the resumes to generate for each document a tensor of the size 512x768. 512 is the number of tokens per document and 768 the number of values for each token. To be able to compare a vector of the job description and one that representing the resume a pooling operation is needed. The idea is to take the mean of all token embeddings and compress them into a single 768 vector space: the result is a sentence vector for each document.

In addition to the functions *get_sentences* and *find_similarity*, which execute the steps described above, two further functions have been implemented. The first one has the name *show_results* and displays the most suitable resumes for a desired job description in numerical descending order. The second function *sort_pdf* is responsible for renaming all resumes that have been checked for similarity with the job description in the folder directory itself, so that a descending order is also recognizable by prefixing numbers in the file name:

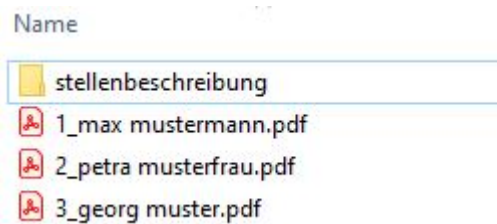


Figure 3.11: Example of directory after running text similarity program

This is to ensure easier operation for the user of the program by visualizing the pdf files of the resumes with the most suitable candidates based on the text similarity.

3.5 Summary

This chapter first explained the structure and functionality of BERT. Thereby, the step for creating feature embeddings was described in more detail using an example for better comprehensibility. Furthermore, the principle of the encoder was described and its two main components: the multi-head attention and the fully connected feed-forward network. Since no new model was created by the pre training process but by means of transfer learning, this approach was explained and the model used for this purpose was presented.

For the evaluation and improving of the individual models the hold-out and leave-one-out procedure was listed as well as, ultimately, all the individual BERT variants and their underlying concept.

Chapter 4

Experiments

The characteristics of the data as well as the procedure for further processing in order to use them with the methods mentioned were presented.

The following chapter shows all relevant results of the experiments which were carried out for the different BERT variants. A total of **158 experiments** were carried out in the context of the different variants and fine tuning measures to find the most suitable models. The chapter primarily presents the results, which serve as a basis for further discussion in chapter 5. The results should ultimately provide information about which hyperparameters had the greatest influence and create a better understanding of the individual solution approaches and their suitability for the evaluation of resumes.

The complete experimental results can be found in appendices A 4 and A5 of this thesis.

4.1 Results of BERT v1-v3 experiments

The first part of the chapter shows the experiments on BERT variants 1-3. The procedure for conducting the experiments is divided into several series of experiments for each of the three Bert variants:

1. in the first series of experiments for a variant, a different batch size and learning rate was tried out for a model. Starting from an average value, lower as well as higher values were used until the results stagnated or became worse.
2. in the second test series of a variant or a model, an exponential learning rate as well as the (un)freezing of the pre-trained bert model and a dropout were tried for the best experiment from the first test series.
3. two more seeds (1 and 40) were tried in addition to seed 100 and the respective epoch with the best results was selected.

In the case of BERT v1, the first test series was created without frozen layers, and after it had become apparent that the activation of frozen layers led to better results, frozen

layers were activated from the beginning for the other BERT variants. All experiments were performed using the hold-out method and afterwards for each best result the loss, accuracy and roc curve (due to leave-one-out) graph were plotted.

The primary measures and metrics used to compare the predictive success of each BERT variant are:

1. train and validation loss:

the loss stands for the prediction error and is calculated as the expected value of the 0-1 loss over n examples in the dataset (once for the train data and once for the validation data) [16].

$$ERR_s = \frac{1}{n} \sum_{i=1}^n L(\hat{Y}_i, Y_i)$$

and $L(\cdot)$ is defined as:

$$L(\hat{Y}_i, Y_i) = \begin{cases} 0, & \text{if } \hat{Y}_i = Y_i \\ 1, & \text{if } \hat{Y}_i \neq Y_i \end{cases}$$

2. train and validation accuracy:

for evaluating the quality of the classification model the accuracy is measured. The final result shows the the percentage of misclassification for the binary classification task [12, p. 5].

$$accuracy = \frac{\text{number of correctly classified examples}}{\text{total number of cases}}$$

3. sensitivity and specificity:

sensitivity stands for the proportion of true positives that are correctly recognized. Sensitivity is also synonymous with the use of the term "recall".

$$sensitivity = recall = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}$$

specificity stands for the proportion of true negatives that are correctly recognized [19].

$$specificity = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}}$$

4. precision:

while sensitivity shows in percentage the correctly prediction for the positive class, precision indicates in percentage which positive predictions are correct [19].

$$precision = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false positives}}$$

5. F1 score:

The value designated as "F1" is the average of "recall" and "precision". It allows to describe the overall accuracy of the predictions of a classifier [19].

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Additionally to determine the degree of generalization capability of the best variants, ROC (Receiver Operator Characteristics) curves were plotted using the leave-one-out method. They are also part of the experiment results, presented in this chapter and discussed in the next one. For the reproducibility of the experiments a random seed of 100 was defined, this value was always used if it was not explicitly specified.

4.1.1 BERT v1

Ep.	Drp.	Ba.	LR	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	0	8	$5e^{-4}$	0.69	0.69	0.5	0.45	0.5	0.55	0.5	0.66
30	0	8	$5e^{-5}$	0.001	3.34	1	0.5	0.4	0.6	0.5	0.44
30	0	8	$5e^{-6}$	0.04	0.82	1	0.75	0.6	0.75	0.85	0.7
30	0	8	$5e^{-7}$	0.66	0.69	0.5	0.41	0.3	0.6	0.42	0.35
30	0	4	$5e^{-4}$	0.6938	0.693	0.5	0.5	0.5	0.52	0.5	0.66
30	0	4	$5e^{-5}$	0.0004	3.4	1	0.55	0.4	0.7	0.57	0.47
30	0	4	$5e^{-6}$	0.007	1.1	1	0.75	0.6	0.75	0.85	0.7
30	0	4	$5e^{-7}$	0.65	0.7	0.68	0.45	0.2	0.4	0.4	0.26
30	0	2	$5e^{-4}$	0.69	0.69	0.48	0.5	0.5	0.48	0.5	0.66
30	0	2	$5e^{-5}$	0.0002	3.6	1	0.55	0.6	0.5	0.54	0.57
30	0	2	$5e^{-6}$	0.002	1.2	1	0.8	0.7	0.75	0.875	0.77
30	0	2	$5e^{-7}$	0.66	0.7	0.68	0.41	0.3	0.5	0.375	0.33

Table 4.1: Varying batch size and learning rate (BERT v1)

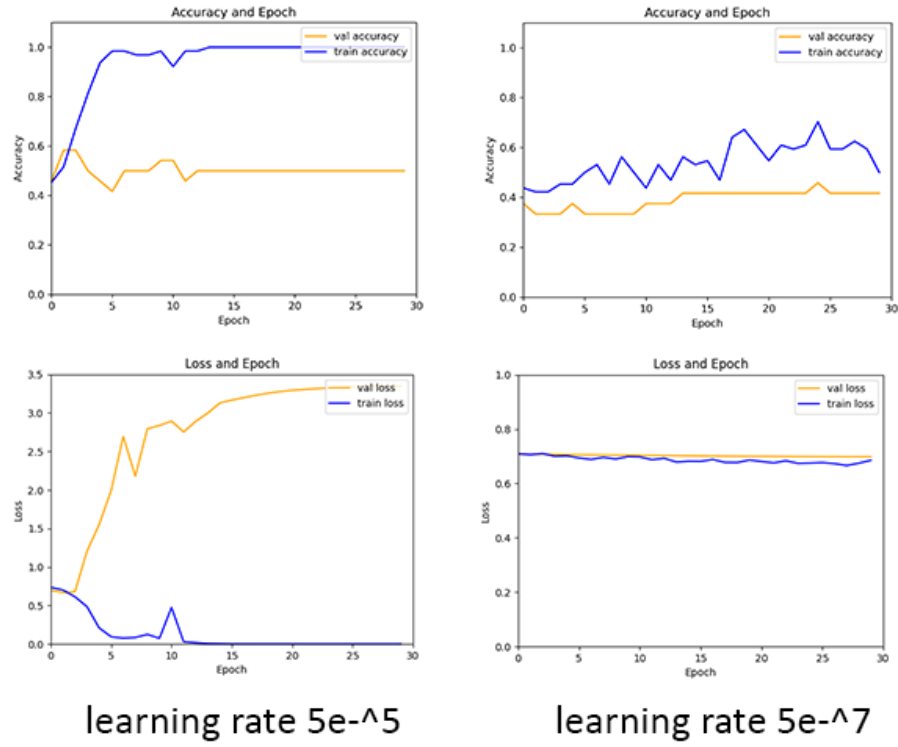


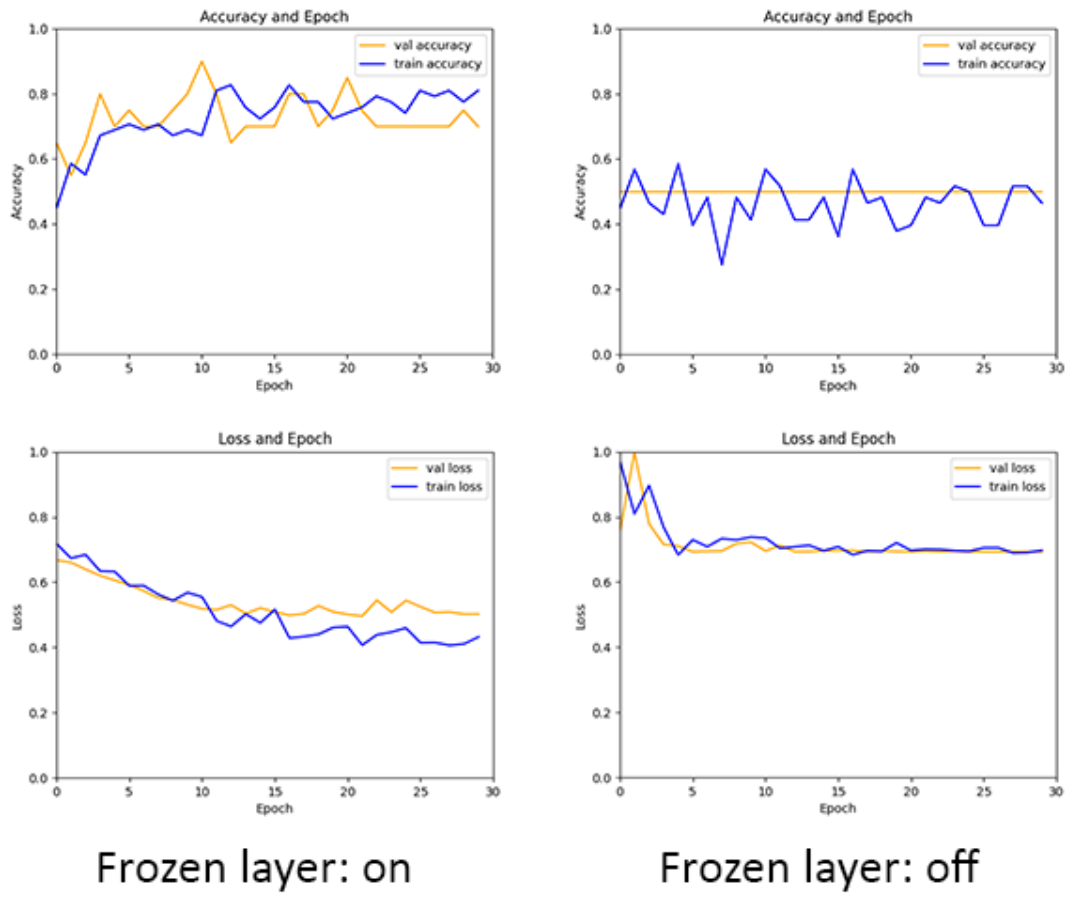
Figure 4.1: Overfitting and underfitting (All layers activated)

Ep.	Ba.	LR	Drp.	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	2	$5e^{-6}$	0	off	off	0.002	1.2	1	0.8	0.7	0.75	0.875	0.77
30	2	$5e^{-6}$	0	on	off	0.69	0.7	0.45	0.45	0.7	0.2	0.46	0.56
30	2	$5e^{-6}$	0	off	on	0.71	0.7	0.4	0.35	0.2	0.35	0.28	0.23
30	2	$5e^{-6}$	0	on	on	0.71	0.7	0.4	0.4	0.2	0.4	0.33	0.25
30	2	$5e^{-6}$	0.1	off	off	0.01	1.03	1	0.8	0.7	0.75	0.875	0.77

Table 4.2: LR $5e^{-6}$ with and without exponential learning rate, frozen layers, dropout (BERT v1)

Ep.	Ba.	LR	Drp.	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	2	$5e^{-4}$	0	off	off	0.69	0.69	0.48	0.5	0.5	0.48	0.5	0.66
30	2	$5e^{-4}$	0	on	off	0.69	0.69	0.5	0.5	0.5	0.5	0.25	0.33
30	2	$5e^{-4}$	0	off	on	0.47	0.51	0.74	0.75	0.6	0.75	0.77	0.74
30	2	$5e^{-4}$	0	on	on	0.69	0.68	0.55	0.55	0.5	0.6	0.55	0.55
30	2	$5e^{-4}$	0.1	off	off	0.69	0.69	0.5	0.5	0.5	0.5	0.25	0.33

Table 4.3: LR $5e^{-4}$ with and without exponential learning rate, frozen layers, dropout (BERT v1)

Figure 4.2: Comparison frozen layer on and off (learning rate $5e^{-4}$)

Ep.	Drp.	Ba.	LR	FL	Seed	LR Exp.	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
13	0	2	$5e^{-4}$	on	10	off	0.54	0.53	0.62	0.70	0.60	0.60	0.71	0.70
9	0	2	$5e^{-4}$	on	40	off	0.4893	0.4886	0.76	0.75	0.70	0.70	0.77	0.70
14	0	2	$5e^{-4}$	on	100	off	0.50	0.50	0.69	0.70	0.70	0.65	0.71	0.70

Table 4.4: Best experiment with different seeds (BERT v1)

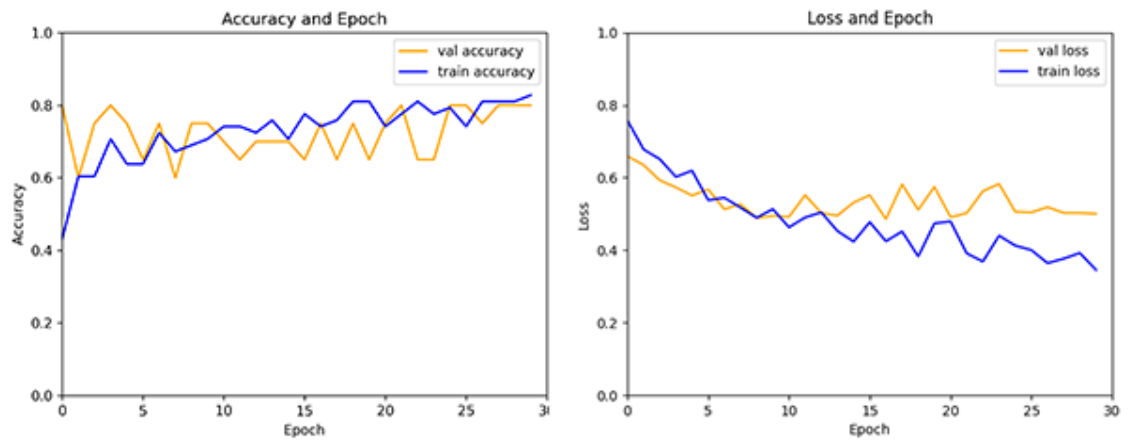


Figure 4.3: Best result: $5e^{-4}$, FL on, seed 40, LR Exp. off (BERT v1)

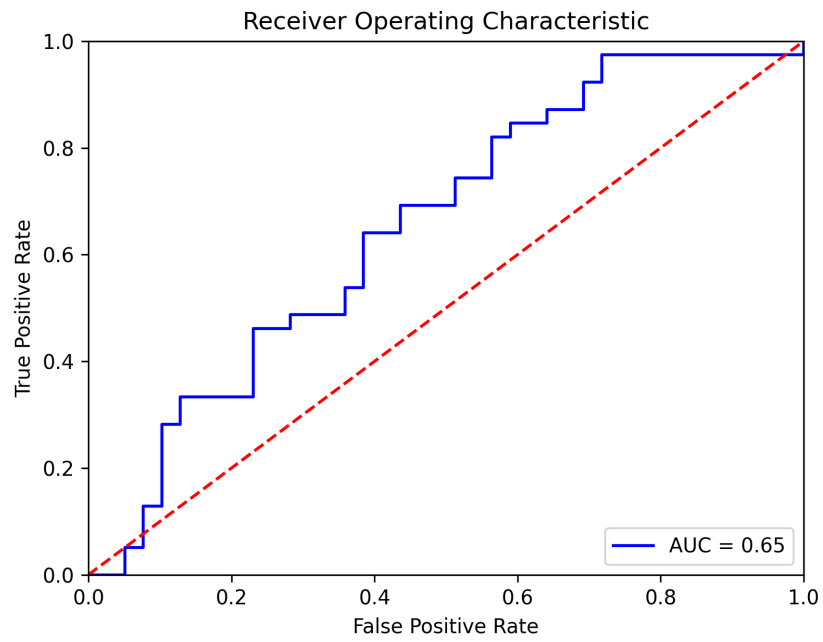


Figure 4.4: ROC Curve (BERT v1)

4.1.2 BERT v2

4.1.2.1 Submodel school

Ep.	Drp.	Ba.	LR	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	0	8	$5e^{-3}$	0.10	0.50	0.98	0.83	0.70	0.50	0.86	0.72
30	0	8	$5e^{-4}$	0.41	0.55	0.8	0.79	0.70	0.57	0.70	0.69
30	0	8	$5e^{-5}$	0.64	0.68	0.61	0.54	0.50	0.55	0.82	0.76
30	0	8	$5e^{-6}$	0.69	0.69	0.61	0.33	0.45	0.42	0.47	0.46
30	0	4	$5e^{-3}$	0.10	0.84	0.96	0.75	0.65	0.42	0.72	0.66
30	0	4	$5e^{-4}$	0.34	0.50	0.87	0.70	0.70	0.42	0.63	0.61
30	0	4	$5e^{-5}$	0.63	0.63	0.62	0.65	0.50	0.50	0.82	0.76
30	0	4	$5e^{-6}$	0.69	0.68	0.55	0.45	0.45	0.57	0.53	0.53
30	0	2	$5e^{-3}$	0.12	0.97	0.96	0.66	0.65	0.42	0.72	0.66
30	0	2	$5e^{-4}$	0.34	0.59	0.85	0.61	0.60	0.57	0.64	0.64
30	0	2	$5e^{-5}$	0.60	0.59	0.65	0.72	0.50	0.60	0.82	0.76
30	0	2	$5e^{-6}$	0.67	0.68	0.63	0.55	0.50	0.60	0.61	0.59

Table 4.5: Varying batch size and learning rate (BERT v2 school submodel)

Ep.	Ba.	LR	Drp.	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	8	$5e^{-4}$	0	off	on	0.41	0.55	0.80	0.79	0.69	0.57	0.70	0.69
30	8	$5e^{-4}$	0	on	on	0.65	0.69	0.62	0.50	0.75	0.50	0.79	0.70
30	8	$5e^{-4}$	0.1	off	on	0.44	0.62	0.79	0.45	0.64	0.57	0.64	0.64
30	8	$5e^{-4}$	0	on	off	0.69	0.71	0.49	0.29	0.35	0.40	0.21	0.29
30	8	$5e^{-4}$	0	off	off	0.70	0.76	0.51	0.29	0.35	0.40	0.21	0.29

Table 4.6: LR $5e^{-4}$ with and without exponential learning rate, frozen layers, dropout (BERT v2 school submodel)

Ep.	Drp.	Ba.	LR	Seed	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
12	0	8	$5e^{-4}$	10	off	on	0.5292	0.5361	0.77	0.79	0.69	0.57	0.70	0.69
09	0	8	$5e^{-4}$	40	off	on	0.5404	0.5484	0.77	0.79	0.71	0.71	0.70	0.70
14	0	8	$5e^{-4}$	100	off	on	0.5198	0.5199	0.73	0.79	0.73	0.69	0.73	0.71

Table 4.7: Best experiment with different seeds (BERT v2 school submodel)

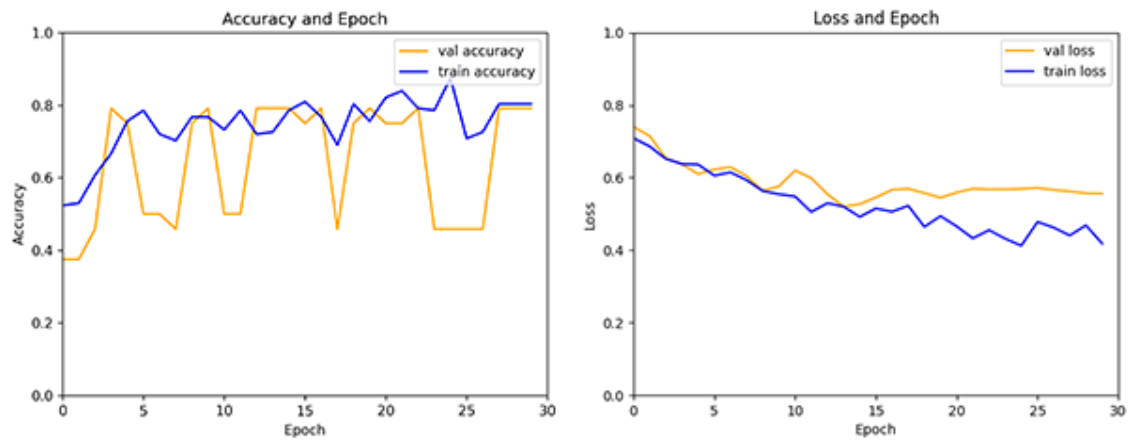


Figure 4.5: Best result: batch size 8, $5e^{-4}$, FL on, seed 100, LR Exp. off (BERT v2 school submodel)

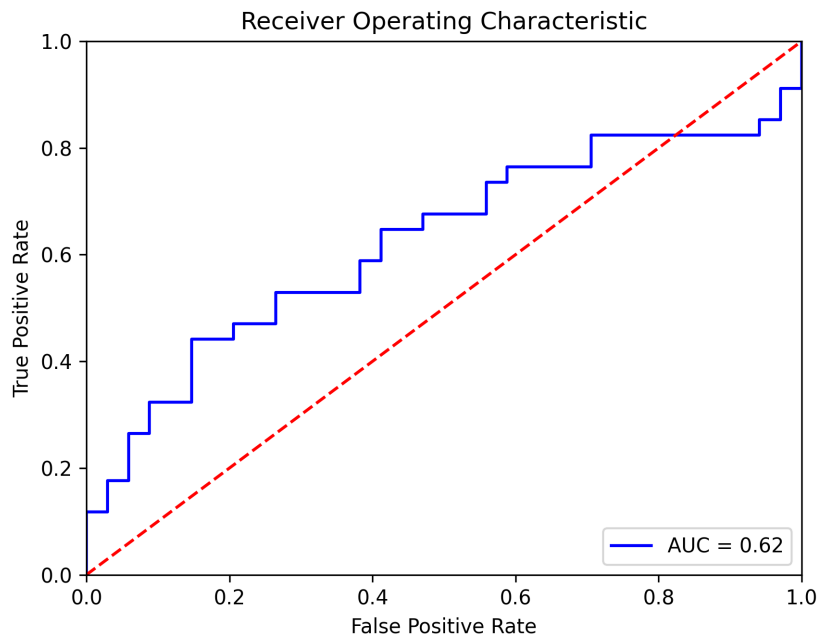


Figure 4.6: ROC Curve (BERT v2 school submodel)

4.1.2.2 Submodel profession

Ep.	Drp.	Ba.	LR	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	0	8	$5e^{-3}$	0.10	1.62	0.97	0.48	0.625	0.45	0.54	0.53
30	0	8	$5e^{-4}$	0.36	0.89	0.88	0.44	0.625	0.36	0.49	0.47
30	0	8	$5e^{-5}$	0.65	0.70	0.70	0.36	0.75	0.09	0.35	0.32
30	0	8	$5e^{-6}$	0.70	0.70	0.52	0.55	0.625	0.45	0.54	0.53
30	0	4	$5e^{-3}$	0.13	1.75	0.95	0.51	0.625	0.45	0.54	0.53
30	0	4	$5e^{-4}$	0.31	0.92	0.87	0.46	0.625	0.34	0.49	0.47
30	0	4	$5e^{-5}$	0.62	0.70	0.73	0.41	0.75	0.18	0.45	0.39
30	0	4	$5e^{-6}$	0.69	0.70	0.52	0.53	0.625	0.45	0.54	0.53
30	0	2	$5e^{-3}$	0.11	2.05	0.95	0.55	0.625	0.45	0.54	0.53
30	0	2	$5e^{-4}$	0.27	1.23	0.88	0.50	0.625	0.32	0.49	0.47
30	0	2	$5e^{-5}$	0.59	0.70	0.78	0.55	0.875	0.27	0.61	0.50
30	0	2	$5e^{-6}$	0.69	0.70	0.57	0.40	0.625	0.27	0.44	0.41

Table 4.8: Varying batch size and learning rate (BERT v2 profession submodel)

Ep.	Ba.	LR	Drp.	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	8	$5e^{-6}$	0	off	on	0.70	0.70	0.52	0.55	0.625	0.45	0.54	0.53
30	8	$5e^{-6}$	0	on	on	0.70	0.70	0.45	0.63	0.63	0.62	0.63	0.63
30	8	$5e^{-6}$	0.1	off	on	0.69	0.70	0.61	0.55	0.625	0.47	0.54	0.53
30	8	$5e^{-6}$	0	on	off	0.67	0.70	0.66	0.36	0.625	0.18	0.38	0.35
30	8	$5e^{-6}$	0	off	off	0.05	1.91	0.98	0.44	0.375	0.54	0.46	0.46

Table 4.9: LR $5e^{-6}$ with and without exponential learning rate, frozen layers, dropout (BERT v2 profession submodel)

Ep.	Drp.	Ba.	LR	Seed	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	0	8	$5e^{-6}$	10	on	on	0.69	0.70	0.44	0.20	0.375	0.18	0.27	0.26
26	0	8	$5e^{-6}$	40	on	on	0.68	0.66	0.55	0.59	0.50	0.50	0.29	0.37
30	0	8	$5e^{-6}$	100	on	on	0.70	0.70	0.45	0.63	0.63	0.62	0.63	0.63

Table 4.10: Best experiment with different seeds (BERT v2 profession submodel)

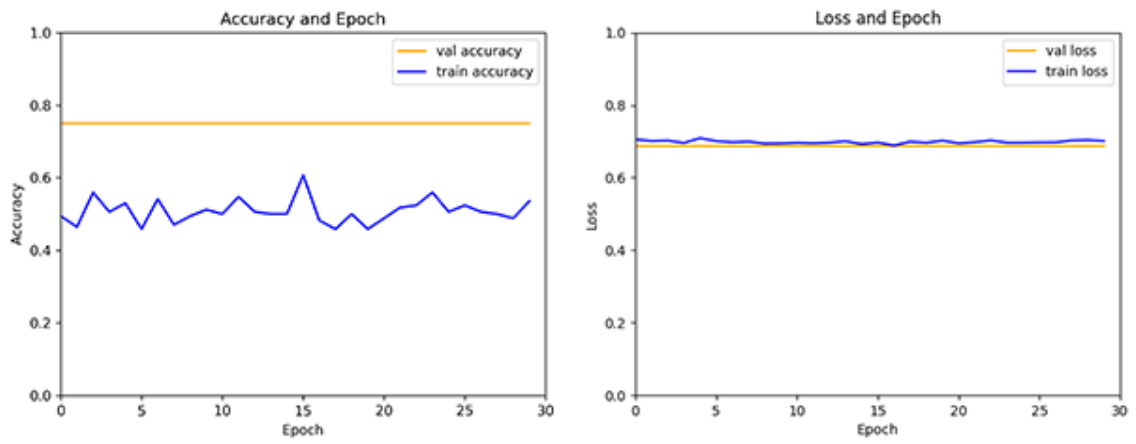


Figure 4.7: Best result: Batchsize 8, $5e^{-6}$, LR Exp. on, FL on, Seed 100 (BERT v2 profession submodel)

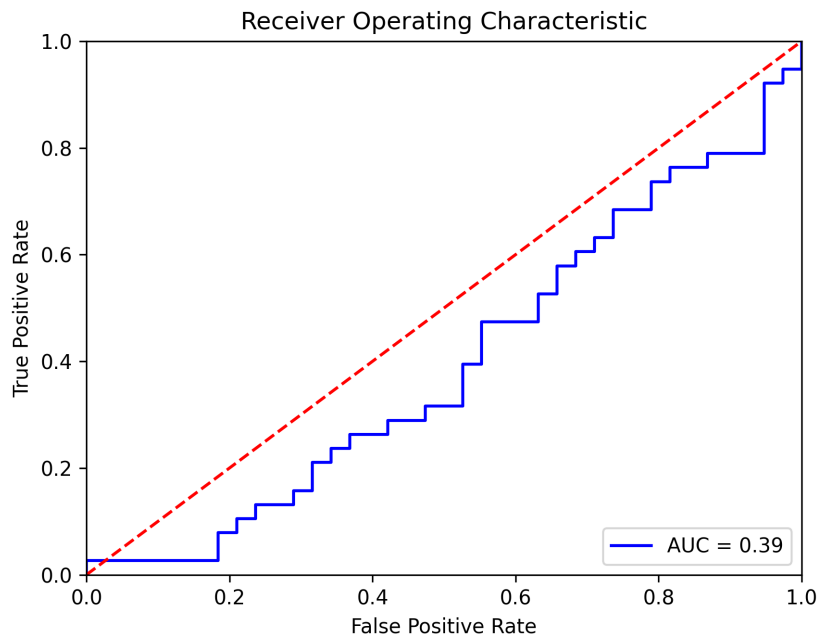


Figure 4.8: ROC Curve (BERT v2 profession submodel)

4.1.2.3 Submodel skills

Ep.	Drp.	Ba.	LR	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	0	8	$5e^{-3}$	0.13	0.93	0.98	0.75	0.77	0.71	0.75	0.75
30	0	8	$5e^{-4}$	0.46	0.59	0.88	0.68	0.66	0.71	0.69	0.69
30	0	8	$5e^{-5}$	0.66	0.70	0.69	0.37	0.11	0.71	0.36	0.33
30	0	8	$5e^{-6}$	0.69	0.70	0.5	0.31	0.55	0.50	0.21	0.24
30	0	4	$5e^{-3}$	0.11	1.08	0.96	0.68	0.66	0.71	0.69	0.69
30	0	4	$5e^{-4}$	0.36	0.54	0.90	0.75	0.77	0.71	0.75	0.75
30	0	4	$5e^{-5}$	0.64	0.69	0.75	0.5	0.55	0.42	0.49	0.49
30	0	4	$5e^{-6}$	0.69	0.70	0.48	0.31	0.5	0.49	0.21	0.24
30	0	2	$5e^{-3}$	0.15	1.18	0.96	0.81	0.88	0.68	0.82	0.81
30	0	2	$5e^{-4}$	0.31	0.56	0.89	0.75	0.77	0.70	0.75	0.75
30	0	2	$5e^{-5}$	0.61	0.69	0.78	0.56	0.55	0.57	0.56	0.56
30	0	2	$5e^{-6}$	0.68	0.70	0.59	0.37	0.55	0.14	0.33	0.33

Table 4.11: Varying batch size and learning rate (BERT v2 skills submodel)

Ep.	Ba.	LR	Drp.	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
7	8	$5e^{-3}$	0	off	on	0.47	0.49	0.83	0.81	0.75	0.71	0.75	0.75
7	8	$5e^{-3}$	0	on	on	0.59	0.65	0.67	0.56	0.53	0.28	0.54	0.52
7	8	$5e^{-3}$	0.1	off	on	0.45	0.57	0.74	0.75	0.77	0.57	0.68	0.68
7	8	$5e^{-3}$	0	on	off	0.76	0.68	0.24	0.56	0.50	0.48	0.28	0.36
7	8	$5e^{-3}$	0	off	off	0.71	0.68	0.41	0.56	0.50	0.50	0.28	0.36

Table 4.12: LR $5e^{-3}$ with and without exponential learning rate, frozen layers, dropout (BERT v2 skills submodel)

Ep.	Drp.	Ba.	LR	Seed	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
8	0	8	$5e^{-3}$	10	off	on	0.42	0.46	0.84	0.81	0.62	0.57	0.62	0.62
2	0	8	$5e^{-3}$	40	off	on	0.68	0.68	0.57	0.50	0.56	0.50	0.73	0.42
7	0	8	$5e^{-3}$	100	off	on	0.47	0.49	0.83	0.81	0.75	0.71	0.75	0.75

Table 4.13: Best experiment with different seeds (BERT v2 skills submodel)

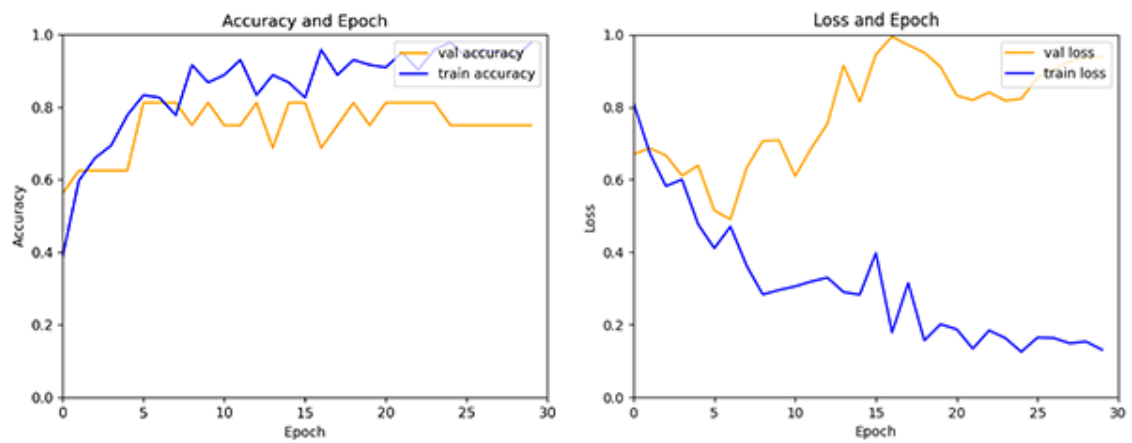


Figure 4.9: Best result: batch size 8, $5e^{-3}$, FL on, seed 100, LR Exp. off (BERT v2 skills submodel)

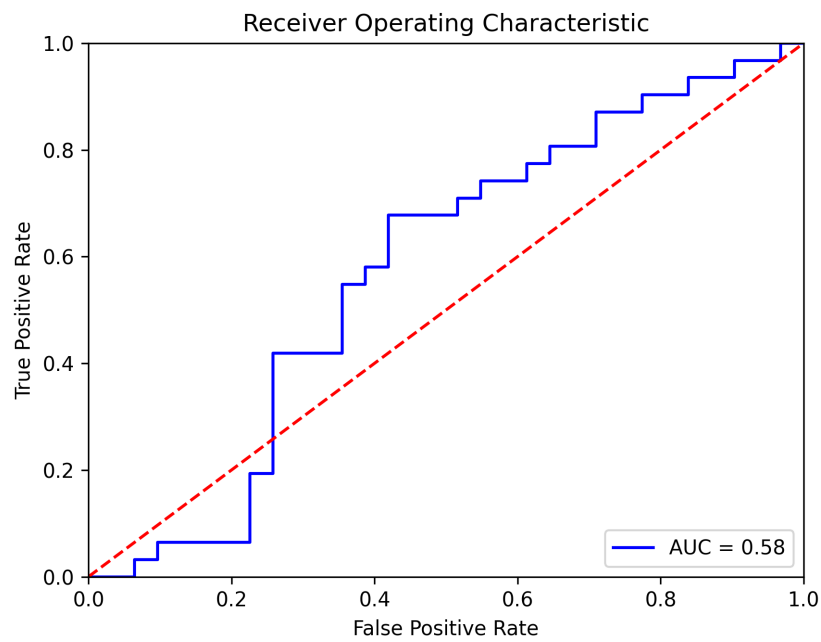


Figure 4.10: ROC Curve (BERT v2 skills submodel)

4.1.3 BERT v3

4.1.3.1 Submodel school

Ep.	Drp.	Ba.	LR	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	0	8	$5e^{-3}$	0.10	0.59	0.96	0.79	0.66	0.42	0.72	0.66
30	0	8	$5e^{-4}$	0.46	0.54	0.81	0.75	0.61	0.32	0.63	0.61
30	0	8	$5e^{-5}$	0.64	0.62	0.66	0.87	0.85	0.625	0.85	0.82
30	0	8	$5e^{-6}$	0.68	0.66	0.60	0.87	0.85	0.85	0.85	0.82
30	0	4	$5e^{-3}$	0.11	0.83	0.96	0.70	0.61	0.42	0.63	0.61
30	0	4	$5e^{-4}$	0.41	0.49	0.84	0.75	0.66	0.46	0.72	0.66
30	0	4	$5e^{-5}$	0.64	0.59	0.61	0.75	0.70	0.71	0.70	0.70
30	0	4	$5e^{-6}$	0.68	0.65	0.59	0.85	0.85	0.85	0.85	0.82
30	0	2	$5e^{-3}$	0.09	0.98	0.96	0.61	0.61	0.40	0.63	0.61
30	0	2	$5e^{-4}$	0.33	0.55	0.87	0.66	0.61	0.38	0.63	0.61
30	0	2	$5e^{-5}$	0.60	0.57	0.75	0.77	0.78	0.80	0.77	0.76
30	0	2	$5e^{-6}$	0.67	0.65	0.63	0.83	0.85	0.60	0.85	0.82

Table 4.14: Varying batch size and learning rate (BERT v3 school submodel)

Ep.	Ba.	LR	Drp.	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	4	$5e^{-4}$	0	off	on	0.41	0.49	0.84	0.75	0.66	0.46	0.72	0.66
30	4	$5e^{-4}$	0	on	on	0.66	0.65	0.65	0.55	0.7	0.57	0.64	0.64
30	4	$5e^{-4}$	0.1	off	on	0.38	0.50	0.88	0.75	0.9	0.42	0.72	0.66
30	4	$5e^{-4}$	0	on	off	0.68	0.71	0.53	0.35	0.5	0.5	0.21	0.29
30	4	$5e^{-4}$	0	off	off	0.68	0.71	0.53	0.35	0.5	0.5	0.21	0.29

Table 4.15: LR $5e^{-4}$ with and without exponential learning rate, frozen layers, dropout (BERT v3 school submodel)

Ep.	Drp.	Ba.	LR	Seed	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
12	0	4	$5e^{-4}$	10	off	on	0.49	0.51	0.81	0.80	0.69	0.57	0.70	0.69
9	0	4	$5e^{-4}$	40	off	on	0.49	0.51	0.81	0.75	0.71	0.71	0.70	0.70
11	0	4	$5e^{-4}$	100	off	on	0.50	0.50	0.76	0.8	0.73	0.70	0.73	0.71

Table 4.16: Best experiment with different seeds (BERT v3 school submodel)

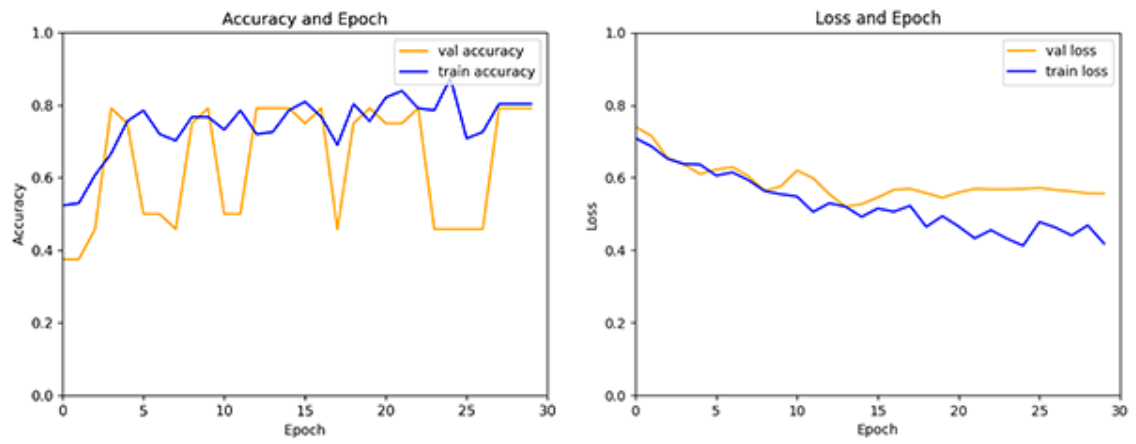


Figure 4.11: Best result: batch size 4, $5e^{-4}$, FL on, seed 100, LR Exp. off (BERT v3 school submodel)

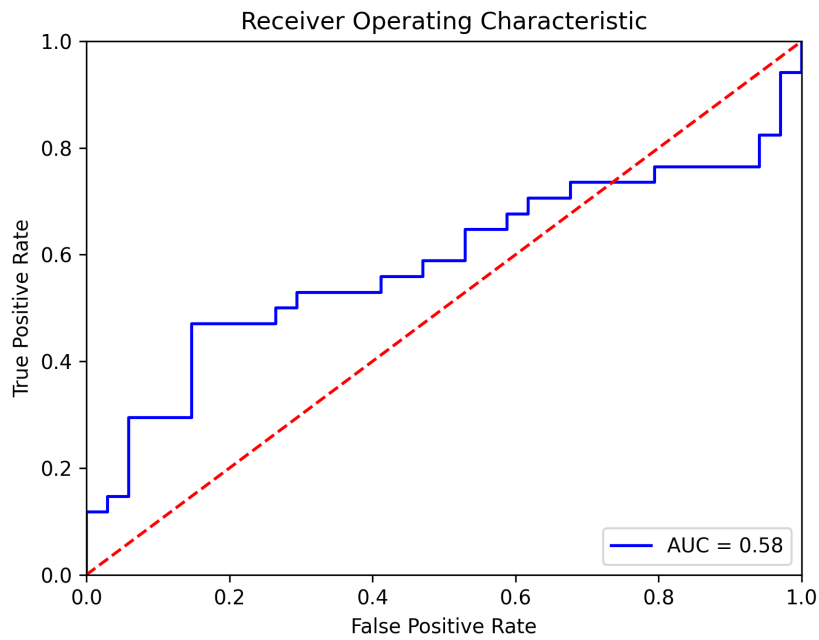


Figure 4.12: ROC Curve (BERT v3 school submodel)

4.1.3.2 Submodel profession

Ep.	Drp.	Ba.	LR	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	0	8	$5e^{-3}$	0.09	1.63	1	0.48	0.625	0.45	0.54	0.53
30	0	8	$5e^{-4}$	0.36	0.89	0.88	0.44	0.625	0.36	0.49	0.47
30	0	8	$5e^{-5}$	0.65	0.70	0.70	0.36	0.75	0.09	0.35	0.32
30	0	8	$5e^{-6}$	0.70	0.70	0.52	0.55	0.625	0.42	0.54	0.53
30	0	4	$5e^{-3}$	0.13	1.75	0.95	0.51	0.625	0.45	0.54	0.53
30	0	4	$5e^{-4}$	0.31	0.92	0.87	0.46	0.625	0.36	0.49	0.47
30	0	4	$5e^{-5}$	0.62	0.70	0.73	0.41	0.75	0.18	0.45	0.39
30	0	4	$5e^{-6}$	0.69	0.70	0.52	0.53	0.625	0.43	0.54	0.53
30	0	2	$5e^{-3}$	0.11	2.05	0.95	0.55	0.625	0.45	0.54	0.53
30	0	2	$5e^{-4}$	0.27	1.23	0.88	0.50	0.625	0.36	0.49	0.47
30	0	2	$5e^{-5}$	0.59	0.70	0.78	0.55	0.875	0.25	0.61	0.50
30	0	2	$5e^{-6}$	0.69	0.70	0.57	0.40	0.625	0.27	0.44	0.41

Table 4.17: Varying batch size and learning rate (BERT v3 profession submodel)

Ep.	Ba.	LR	Drp.	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	8	$5e^{-6}$	0	off	on	0.70	0.70	0.52	0.55	0.625	0.42	0.54	0.53
30	8	$5e^{-6}$	0	on	on	0.70	0.70	0.41	0.60	0.63	0.60	0.63	0.63
30	8	$5e^{-6}$	0.1	off	on	0.69	0.70	0.53	0.40	0.63	0.27	0.44	0.41
30	8	$5e^{-6}$	0	on	off	0.69	0.69	0.53	0.50	0.63	0.45	0.54	0.53
30	8	$5e^{-6}$	0	off	off	0.07	2.76	0.98	0.50	0.375	0.54	0.46	0.46

Table 4.18: LR $5e^{-6}$ with and without exponential learning rate, frozen layers, dropout (BERT v3 profession submodel)

Ep.	Drp.	Ba.	LR	Seed	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	0	8	$5e^{-6}$	10	on	on	0.69	0.70	0.48	0.20	0.375	0.18	0.27	0.26
30	0	8	$5e^{-6}$	40	on	on	0.68	0.67	0.59	0.59	0.5	0.5	0.29	0.37
30	0	8	$5e^{-6}$	100	on	on	0.70	0.70	0.41	0.60	0.63	0.60	0.63	0.63

Table 4.19: Best experiment with different seeds (BERT v3 profession submodel)

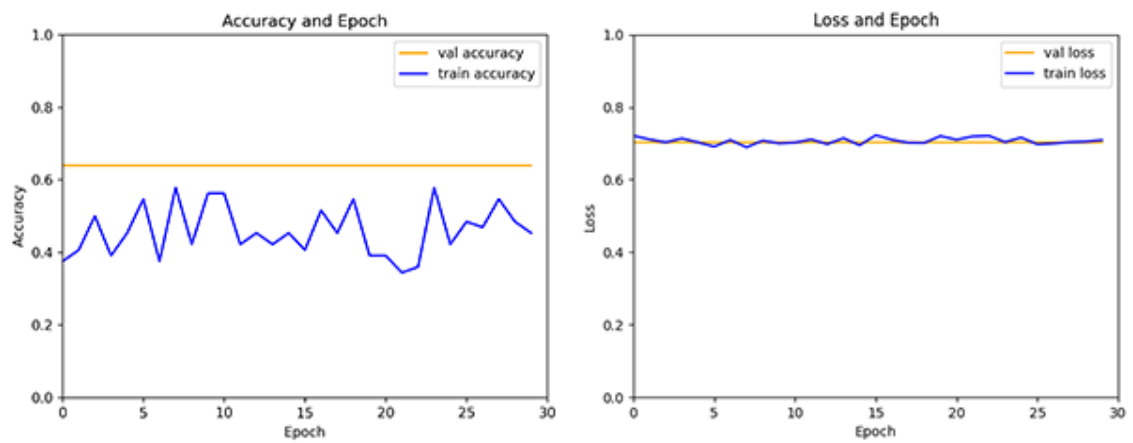


Figure 4.13: Best result: batch size 8, $5e^{-6}$, FL on, seed 100, LR Exp. on (BERT v3 profession submodel)

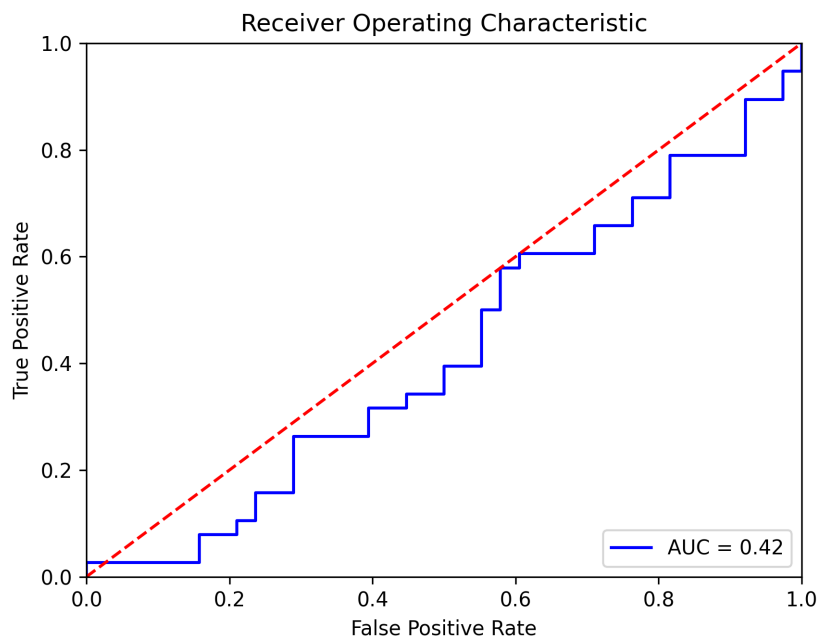


Figure 4.14: ROC Curve (BERT v3 profession submodel)

4.1.3.3 Submodel skills

Ep.	Drp.	Ba.	LR	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	0	8	$5e^{-3}$	0.13	0.93	0.98	0.75	0.77	0.71	0.75	0.75
30	0	8	$5e^{-4}$	0.46	0.59	0.88	0.68	0.66	0.71	0.69	0.69
30	0	8	$5e^{-5}$	0.66	0.70	0.69	0.37	0.11	0.18	0.36	0.33
30	0	8	$5e^{-6}$	0.69	0.70	0.30	0.31	0.28	0.30	0.21	0.24
30	0	4	$5e^{-3}$	0.11	1.08	0.96	0.68	0.66	0.74	0.69	0.69
30	0	4	$5e^{-4}$	0.36	0.54	0.90	0.75	0.77	0.71	0.75	0.75
30	0	4	$5e^{-5}$	0.64	0.69	0.75	0.50	0.55	0.42	0.49	0.49
30	0	4	$5e^{-6}$	0.69	0.70	0.48	0.31	0.28	0.52	0.21	0.24
30	0	2	$5e^{-3}$	0.15	1.18	0.96	0.81	0.88	0.71	0.82	0.81
30	0	2	$5e^{-4}$	0.31	0.56	0.89	0.75	0.77	0.70	0.75	0.75
30	0	2	$5e^{-5}$	0.61	0.69	0.78	0.56	0.55	0.57	0.56	0.56
30	0	2	$5e^{-6}$	0.67	0.70	0.59	0.37	0.11	0.14	0.33	0.36

Table 4.20: Varying batch size and learning rate (BERT v3 skills submodel)

Ep.	Ba.	LR	Drp.	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	8	$5e^{-6}$	0	off	on	0.69	0.70	0.50	0.31	0.28	0.30	0.21	0.24
30	8	$5e^{-6}$	0	on	on	0.69	0.71	0.51	0.31	0.28	0.30	0.21	0.24
30	8	$5e^{-6}$	0.1	off	on	0.69	0.71	0.48	0.31	0.28	0.29	0.21	0.24
30	8	$5e^{-6}$	0	on	off	0.66	0.70	0.76	0.37	0.55	0.25	0.33	0.33
30	8	$5e^{-6}$	0	off	off	0.20	0.53	0.98	0.81	0.88	0.71	0.82	0.81

Table 4.21: LR $5e^{-6}$ with and without exponential learning rate, frozen layers, dropout (BERT v3 skills submodel)

Ep.	Drp.	Ba.	LR	Seed	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
30	0	8	$5e^{-6}$	10	off	on	0.70	0.69	0.51	0.56	0.28	0.30	0.28	0.36
30	0	8	$5e^{-6}$	40	off	on	0.71	0.73	0.51	0.43	0.28	0.30	0.22	0.30
30	0	8	$5e^{-6}$	100	off	on	0.69	0.70	0.50	0.31	0.28	0.30	0.21	0.24

Table 4.22: Best experiment with different seeds (BERT v3 skills submodel)

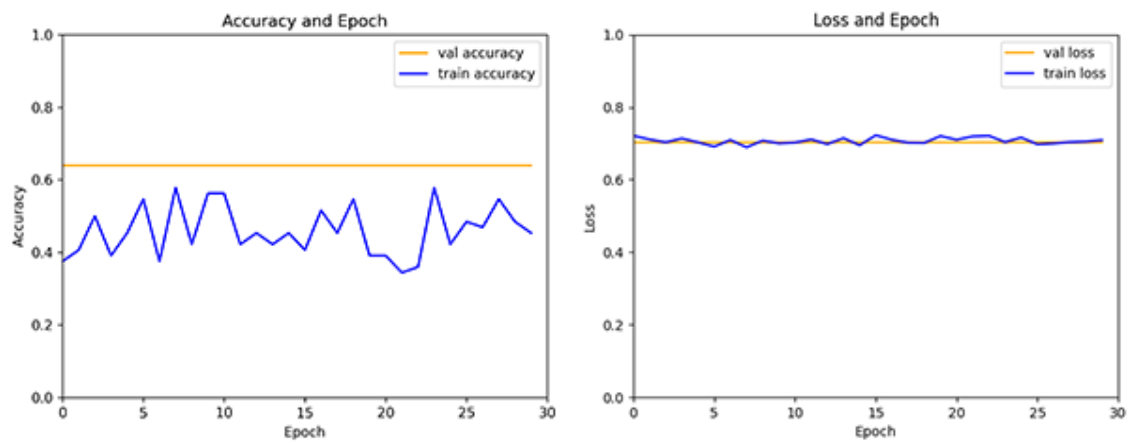


Figure 4.15: Best result: batch size 8, $5e^{-6}$, FL on, seed 10, LR Exp. off (BERT v3 skills submodel)

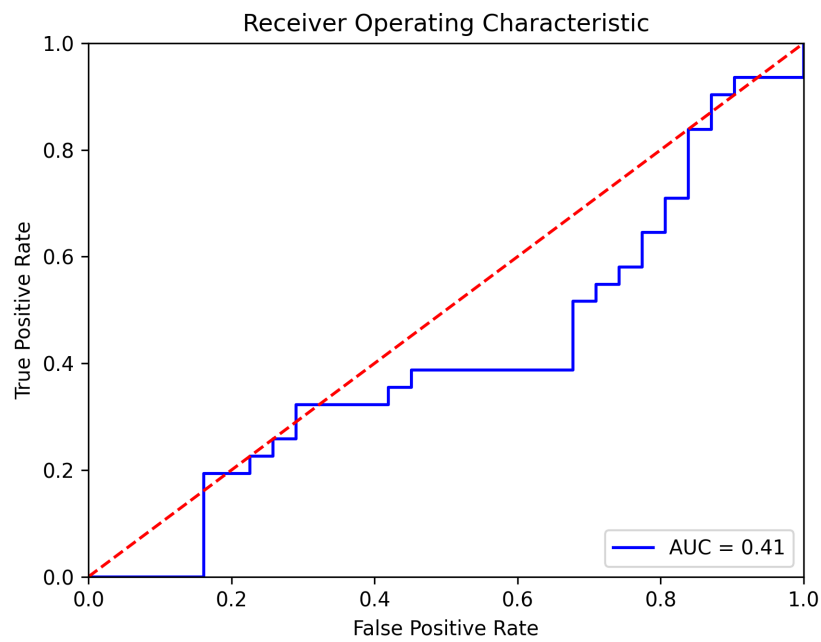


Figure 4.16: ROC Curve (BERT v3 skills submodel)

4.1.4 BERT v1-v3 best results

	Ep.	Drp.	B.	LR	Seed	LR Exp.	FL	T. Loss	V. Loss	T. Acc.	V. Acc.	Sen.	Spec.	Prec.	F1
1	9	0	2	$5e^{-4}$	40	off	on	0.4893	0.4886	0.76	0.75	0.70	0.70	0.71	0.70
2	14	0	8	$5e^{-4}$	100	off	on	0.5198	0.5199	0.73	0.79	0.73	0.69	0.73	0.71
3	30	0	8	$5e^{-6}$	100	on	on	0.70	0.70	0.45	0.63	0.63	0.62	0.63	0.63
4	7	0	8	$5e^{-3}$	100	off	on	0.47	0.49	0.83	0.81	0.75	0.71	0.75	0.75
5	11	0	4	$5e^{-4}$	100	off	on	0.50	0.50	0.76	0.80	0.73	0.70	0.73	0.71
6	30	0	8	$5e^{-6}$	100	on	on	0.70	0.70	0.41	0.60	0.63	0.60	0.63	0.63
7	30	0	8	$5e^{-6}$	10	off	on	0.70	0.69	0.51	0.56	0.28	0.30	0.28	0.36

Table 4.23: Best Results of BERT v1, v2 and v3 experiments

Row	Model	AUC
1	BERT v1	0.65
2	school v2	0.62
3	prof v2	0.39
4	skills v2	0.58
5	school v3	0.58
6	prof v3	0.42
7	skills v3	0.41

Table 4.24: Assignment of the row number to the model name

In addition, ray tune was implemented and adapted to try out hyperparameter tuning for the BERT models under parallel GPU usage (using population based training). Ray tune is a Python library for running experiments and tuning hyperparameters. However, this method turned out to be too time consuming due to the high diversity of hyper parameters.

4.2 Results of BERT v4 experiments

For the experimentation and evaluation process of BERT v4, three test series were carried out for each of two different job profiles (quality planner and injection molding production manager). However, a different structure was chosen for this, as the variant is essentially different from the previous three. As already mentioned in the chapter on methods, this is not, strictly speaking, a binary classification problem. Three pre-trained Bert models were used for the experiments:

1. base-cased-german from huggingface (which was also used for the previous variants).

2. the BERT V1 model with the best results, which was also used as the starting model for the BERT v2 submodels.

3. base-multilingual-uncased was additional used to receive more comprehensive results.

Furthermore, the corresponding label of each resume that was tested for similarity to the job description is listed, as well as the percentage of similarity and the file name of the resume (as a file number) to see if the different models ranked the same files or others as very or less similar.

4.2.1 Quality planner

Job profile	Model	Previous labeled as	Similarity	File number
Qualitätsplaner	base-cased-german	acceptance	0.9424	3
Qualitätsplaner	base-cased-german	rejection	0.9336	6
Qualitätsplaner	base-cased-german	acceptance	0.9267	2
Qualitätsplaner	base-cased-german	rejection	0.8972	7
Qualitätsplaner	base-cased-german	rejection	0.8664	4
Qualitätsplaner	base-cased-german	acceptance	0.8392	1
Qualitätsplaner	base-cased-german	rejection	0.8355	5

Table 4.25: Job profile: Qualitätsplaner and model: base-cased-german

Job profile	Model	Previous labeled as	Similarity	File number
Qualitätsplaner	BERT V1	acceptance	0.9463	3
Qualitätsplaner	BERT V1	rejection	0.9353	6
Qualitätsplaner	BERT V1	acceptance	0.9232	2
Qualitätsplaner	BERT V1	rejection	0.9041	7
Qualitätsplaner	BERT V1	rejection	0.8786	4
Qualitätsplaner	BERT V1	rejection	0.8758	5
Qualitätsplaner	BERT V1	acceptance	0.8737	1

Table 4.26: Job profile: Qualitätsplaner and model: BERT v1

Job profile	Model	Previous labeled as	Similarity	File number
Qualitätsplaner	multilingual	acceptance	0.9179	3
Qualitätsplaner	multilingual	acceptance	0.8952	1
Qualitätsplaner	multilingual	rejection	0.8746	7
Qualitätsplaner	multilingual	rejection	0.8670	5
Qualitätsplaner	multilingual	rejection	0.8593	4
Qualitätsplaner	multilingual	rejection	0.8470	6
Qualitätsplaner	multilingual	acceptance	0.8390	2

Table 4.27: Job profile: Qualitätsplaner and model: multilingual

4.2.2 Production manager injection molding

Job profile	Model	Previous labeled as	Similarity	File number
Fertigungsleiter Spritzguss	base-cased-german	acceptance	0.9409	5
Fertigungsleiter Spritzguss	base-cased-german	acceptance	0.9232	1
Fertigungsleiter Spritzguss	base-cased-german	rejection	0.9187	10
Fertigungsleiter Spritzguss	base-cased-german	acceptance	0.9155	3
Fertigungsleiter Spritzguss	base-cased-german	acceptance	0.9045	2
Fertigungsleiter Spritzguss	base-cased-german	rejection	0.9033	6
Fertigungsleiter Spritzguss	base-cased-german	rejection	0.8779	9
Fertigungsleiter Spritzguss	base-cased-german	acceptance	0.8729	4
Fertigungsleiter Spritzguss	base-cased-german	rejection	0.8671	7
Fertigungsleiter Spritzguss	base-cased-german	rejection	0.7595	8

Table 4.28: Job profile: Fertigungsleiter Spritzguss and model: base-cased-german

Job profile	Model	Previous labeled as	Similarity	File number
Fertigungsleiter Spritzguss	BERT V1	acceptance	0.9471	5
Fertigungsleiter Spritzguss	BERT V1	acceptance	0.9275	3
Fertigungsleiter Spritzguss	BERT V1	acceptance	0.9268	1
Fertigungsleiter Spritzguss	BERT V1	rejection	0.9264	10
Fertigungsleiter Spritzguss	BERT V1	rejection	0.9091	6
Fertigungsleiter Spritzguss	BERT V1	acceptance	0.9077	2
Fertigungsleiter Spritzguss	BERT V1	rejection	0.8989	9
Fertigungsleiter Spritzguss	BERT V1	acceptance	0.8919	4
Fertigungsleiter Spritzguss	BERT V1	rejection	0.8834	7
Fertigungsleiter Spritzguss	BERT V1	rejection	0.8048	8

Table 4.29: Job profile: Fertigungsleiter Spritzguss and model: BERT v1

Job profile	Model	Previous labeled as	Similarity	File number
Fertigungsleiter Spritzguss	multilingual	rejection	0.9190	10
Fertigungsleiter Spritzguss	multilingual	rejection	0.9111	8
Fertigungsleiter Spritzguss	multilingual	acceptance	0.9086	5
Fertigungsleiter Spritzguss	multilingual	acceptance	0.9045	3
Fertigungsleiter Spritzguss	multilingual	rejection	0.9044	9
Fertigungsleiter Spritzguss	multilingual	acceptance	0.8580	4
Fertigungsleiter Spritzguss	multilingual	rejection	0.8538	6
Fertigungsleiter Spritzguss	multilingual	acceptance	0.8453	2
Fertigungsleiter Spritzguss	multilingual	rejection	0.8210	7
Fertigungsleiter Spritzguss	multilingual	acceptance	0.8197	1

Table 4.30: Job profile: Fertigungsleiter Spritzguss and model: multilingual

Chapter 5

Discussion

Based on the results of the experimental data presented in chapter 4, assumptions were made in the following chapter about the individual performance of the models using different hyperparameters and other settings. The results of the best performing models are also compared with each other using the previously presented tables and metrics which are used for the discussion and comparison of the experimental data.

5.1 BERT v1

Looking at the first test series (table 4.1) with varying batch size and learning rate, there are significant outliers for all three batch sizes (2,4 and 8) with the same learning rate of $5e^{-5}$ with respect to the train and validation loss. The experiments with a batch size of 8 and a learning rate of $5e^{-5}$ achieved a train loss of 0.0004% and a validation loss of 3.4%. Here it becomes clear that the model with the chosen hyper parameters tends strongly to overfitting. This overfitting is to be judged as negative, because the actual goodness of fit is obscured and the model is better adapted to the data of the sample, however, it is useless for the classification of new data due to missing generality. With the increase and decrease of the learning rate, the overfitting decreases massively until it no longer occurs, even up to the 30th epoch of an experiment.

Looking at the experiments without overfitting, it can be seen that the models have difficulties to estimate original data as well as new data. These experiments show an underfitting. Since the performance does not necessarily have to fall steadily after the first epochs, an epoch value of 30 was selected for all further experiments and then, in the last test series (table 4.4): "Best experiment with different seeds", the respective epoch was searched for, which shows no overfitting and least amount of underfitting.

In table 4.2, the parameters were taken from the previous series of experiments with the best results (in particular, the individual epochs from 1-30 were also examined and then the hyper parameter selection: Batch size 2 and LR $5e^{-6}$ were selected, since these had the lowest underfitting and no overfitting in previous epochs (figure 4.1)). With additional

activation of the exponential learning rate and/or freezing of the layer of the pre-trained model, the overfitting disappeared, but the level of underfitting was high with a train loss of 71% and a validation loss of 70%.

To better determine the effect of an exponential learning rate, frozen layers, and dropout, the smallest learning rate ($5e^{-4}$) was used to run a series of experiments (table 4.3). The deactivation of the layers of the pre-trained model resulted in a training loss of 47% and a validation loss of 51%. The results of this experiment are therefore the ones with the lowest underfitting and free of significant overfitting. On figure 4.2, the different effects on the accuracy and the loss due to the freezing of the layer can be observed on the one hand, and on the other hand it can be seen that only from the 15th epoch on the train and validation loss are slightly different from each other. Considering the results and these graphs, the arrangement of the hyper parameters from table 4.1 (with frozen layers, without exponential learning rate and dropout) was chosen to select the epoch and seed for the best experiment of BERT v1 in table 4.4.

The best result with a train and validation loss of 48%, a validation accuracy of 75% was found using a seed value of 40 in the 9 epoch. Figure 4.4 shows the roc curve plotted using the leave one out method for the best hyper parameters found. Here, the generalization capability of the model can be seen in more detail. The area under the curve is with 65% not large and at the beginning as well as at the end of the graph it can be seen that for a short time the red diagonal is undercut for these value ranges the model is not able to generalize. The result itself could be better, but considering the small and strongly varying data set, it is still a compromise for the creation of a model for the intended binary classification task.

5.2 BERT v2

Due to the improved results when freezing the pre-trained layers, these were frozen for the further test series with the first table creation "Varying batch size and learning rate" of a sub-model.

5.2.1 Submodel school

The first set of experiments (table 4.5) of the sub model school of BERT v2 shows that with the decrease of the learning rate (independent of the batch size) the train and validation loss values converge and thus the over fitting decreases. However, the validation loss increases at the same time and the model is less able to generalize. For this reason, the results with a learning rate of $5e^{-4}$ was a acceptable compromise. The variant with a batch size of 8 looks

best (training loss 41% and validation loss 55%). A closer look at the individual epochs shows that the loss values only start to improve after further epochs (figure 4.5).

Table 4.6 shows that with exponential learning rate the training and validation loss can be approximated towards the 30 epochs, however, as in the previous test series, this leads to a deterioration of the generalization ability. Further results of the test series were significantly worse. This can be seen when deactivating the exponential learning rate and without freezing the layers. Here, the training loss increases to 70% and the validation loss to 76%. The validation accuracy even drops to 29%.

In the last test series for the sub model school of BERT v2, it becomes clear that the best results can be achieved using the seed value 100, albeit with a slight difference.

5.2.2 Submodel profession

As in the first test series of the previous sub-model, it is evident from the results that overfitting decreases as the learning rate decreases. A batch size of 8 and a learning rate of $5e^{-6}$ led to the best results compared to the selection of other hyper parameter values.

In the following test series with and without exponential learning rate, frozen layers, dropout (table 4.9), the validation accuracy could be increased from 55% to 63% and the precision from 54% to 63% by using an exponential learning rate. When using all layers and deactivating the exponential learning rate, a strong overfitting occurred.

A closer examination of the results of the test series with a different seed did not show any improvement in validation accuracy. On contrary it decreased by 4% with a seed of 40 and even fell to 20% with a seed of 10.

The ROC curve is below the diagonal and the area under the curve is 39%. This means that the sub model profession of BERT v2 is not able to generalize correctly. The reason for this is almost certainly due to the nature of the data provided. On the one hand, the data is small in number, and on the other hand, it is a major hurdle for the BERT model to identify unique features for the binary classification of resumes into rejection and acceptance. The difficulty of identifying features that are relevant to the resumes is problematic because, as already mentioned in chapter 2, 23 different job profiles can be found in the data set and each job profile varies in its criteria for a rejection or acceptance.

5.2.3 Submodel skills

When examining the first test series, it can be seen that with a decreasing learning rate, a reduction in the overfitting occurs for all three batch sizes, but the underfitting also increases. For this reason, the individual epochs of all experiments were examined and it

was found that for the experiment of the first series, i.e.: batchsize 8 and learning rate $5e^{-3}$, the training loss is 47% and the validation loss is 49% in epoch 7. (figure 4.9) In the following epochs, the train and validation loss graphs start to diverge rapidly and overfitting occurs. For the values batchsize 8 and learning rate $5e^{-4}$ the results in epoch 30 also seemed to be quite good and it was important to take a look on previous epochs (figure 4) but from the beginning the overfitting has occurred.

Using exponential learning rate, dropout or activating a layer, all results deteriorate. In the test series with a varying seed no improvement of the resolute could be achieved, therefore the previously selected seed of 100 is the most suitable.

The graph of the ROC curve is initially below the red diagonal line, indicating that the model had difficulty generalizing at the beginning of the leave one out evaluation. However, the ROC curve rises above the diagonal line and then runs above it until the end of the evaluation.

5.3 BERT v3

After discussing the test series of the individual submodels of BERT v2, the results of the submodels of BERT v3 are presented in more detail. In the chapter 5.4, the totality of the best results is compared and discussed.

5.3.1 Sub-model school

As has often been observed in previous series of experiments, it is also the case in the series of experiments (table 4.14) that with a reduction in the learning rate the over fitting decreases, but at the same time the underfitting also increases. The three results with a learning rate of $5e^{-4}$ seem to be a good compromise at first sight (for more certainty, the individual epochs of the individual experiments were considered in order to exclude the possibility that the results are better at an earlier epoch of another variant - see figure 5).

It turned out that here the experiment with a batchsize of 4 is most suitable due to the smallest delta (8%) for training and validation loss in contrast to the experiments with the same learning rate and a batchsize of 2 (train validation loss delta 22%) and 8 (train validation loss delta 8%).

The series of experiments in table 4.15 showed that there was no improvement when activating the exponential rate (increases training loss by 25% and validation loss by 16%) or the dropout (leads to stronger overfitting) and also when deactivating the layer of the pre-trained model. For this reason, the best experiment (batch size 4, learning rate $5e^{-4}$)

from the previous series of experiments was further used to explore the effect of another seed. The choice of a different seed did not show any remarkable influence on the results.

The ROC curve illustrates that the sub model school of the bert v3 is close to the range where the model would be considered unable to generalize.

5.3.2 Sub-model profession

In the "varying batch size and learning rate" series of experiments (4.17), a learning rate of $5e^{-5}$ and $5e^{-6}$ is shown to be the most suitable in terms of eliminating overfitting. train loss 70% validation loss 70%. Due to the 19% higher validation accuracy and precision compared to the variant with a learning rate of $5e^{-5}$, the learning rate of $5e^{-6}$ and batch size 8 was chosen for the further test series.

In table 4.18, the further experiments resulted in minimal changes with respect to the train and validation loss. Except for the experiment where all layers were activated. Here the validation loss increased to 276% and the train loss decreased to 7% which means that the model is overfitting and unsuitable for the classification of new resumes.

When the exponential learning rate was activated, the validation accuracy increased by 5% and the precision by 9%, which is why this was used for the further determination of the best result in table 4.19. In the last series of tests of the sub-model profession of BERT v3 it was shown that the change of the seed in both cases led to a significant deterioration of the validation accuracy (up to 40%), the sensitivity (up to 25%), specificity (up to 45%), precision (up to 36%) and the F1 score (up to 37%).

Analogous to the presented ROC curve of the sub-model profession of BERT v2, the same conclusion can be drawn with regard to the lack of generalization capability of the model.

5.3.3 Submodel skills

Considering the first series of experiments, a batch size of 8 and a learning rate of $5e^{-4}$ with a train loss of 46% and validation loss of 59% seems to be a potential candidate (assuming that in earlier epochs the overfitting is smaller or does not occur at all and that the loss is not significantly higher than in the 30th epoch). However, when examining the individual epochs, it can be seen (figure .2) that overfitting already occurs from the first epochs and thus the hyper parameter choice must be classified as rather unsuitable. Thus the following test series was continued with the batch size 8 and learning rate of $5e^{-6}$ variant.

The test series in table (4.21) did not show any improvements. However, there was a significant deterioration when using all layers. It is also interesting that despite the strong

overfitting, the F1 score and the precision stand out as the highest values. This is due to the fact that these figures were calculated independently of the loss.

In the last series of tests, a seed with a value of 10 was used to increase the validation accuracy from 31% (with a seed value of 100) to as much as 56%.

With regard to the ROC curve, the sub-model is not able to generalize meaningfully.

5.4 Comparison of BERT v1, v2 and v3 results

When comparing the best results of all models (table 4.23 and table 4.24), several interesting findings can be made.

The BERT V1 model is the best at generalization with an AUC value of 65% closely followed by the school v2 model with 62%. Furthermore, the BERT v1 model with a training and validation loss of 48% shows the best results in terms of avoiding overfitting and at the same time minimizing underfitting.

With regard to the results of the BERT v2 and v3 submodels, it is clear that both school submodels deliver similarly good results:

- sub-model school v2: training and validation loss of 51% and AUC value of 62%.
- sub model school v3: training and validation loss of 50% and AUC value of 58%

for the two sub models prof v2 and v3 the results are much worse and also quite similar to each other:

- sub model prof v2: train and validation loss of 70% and AUC value of 39%.
- sub model prof v3: training and validation loss of 70% and AUC value of 42%.

to be precise, these two sub-models are not suitable for the binary classification of resumes, because they are not able to generalize correctly. As already mentioned in 5.2.2, this is probably mainly due to the fact that the number of files is small (78 records) and that they contain 23 different job profiles. Most of the occupations have other relevant criteria in terms of professional experience or history, which makes it difficult to identify uniform features.

Surprisingly, the two results between the skills sub-models v2 and v3 are different compared to the previous sub-models:

- sub model skills v2: training loss of 47% and validation loss of 49% and AUC value of 0.58%.
- sub model skills v3: training loss of 69% and validation loss of 70% as well as AUC value of 41%.

In terms of validation accuracy, the submodel skills v2 with 81% and the submodel school v3 with 80% are the models with the highest value. In general, when determining the validation accuracy, it was also noticed that the value turned out to be higher in some cases when the delta of the training and validation loss increased over several epochs (this can be seen in more detail, for example, in Figure 4.9).

The value ranges of sensitivity, specificity, precision as well as the F1 score are in the range of 60-75% with the exception of the results of the sub-model skills of BERT v3 which performs significantly worse with regard to the metrics.

5.5 BERT v4

The order of the file numbers in the test series (table 4.25 and table 4.26) for the base-cased german and BERT v1 models are the same for the job advertisement with the quality planner job profile, with the exception of one element. In the comparison of the two models for the job profile production manager injection molding, however, the deviation is 6 in 10 resumes. It is interesting to see that the top 50%, i.e. the 5 resumes that have been classified most similarly, when using the base-cased-german model consist to 80% of resumes that belong to the dataset with the label "acceptance". When the BERT V1 model is used, only 60% of the resumes belong to the resumes with the label "acceptance", whereas in the base-cased-german model only the first two resumes belong to the resumes with the label "acceptance".

In the case of the occupational profile quality planner, the test series of all three used models show that 2 of the 3 resumes, which correspond to the top 50% of analyzed texts, belong to the labeled data set "acceptance". On the basis of the experimental data available, the base-cased german model is thus to be classified as the most suitable for achieving an effective preselection of the applications according to the quality criterion of text similarity.

Chapter 6

Conclusion

The purpose of this thesis was to develop and test various models based on BERT in order to provide the Dr. Schneider Unternehmensgruppe with an intelligent matching procedure for internal personnel selection.

For this the resume data was first converted, anonymized and analyzed in order to further optimize its nature for the use of machine learning. The labeled resume data was then provided in a processable format by implementing program code for feature embedding for the training and evaluation processes. Three different variants were experimentally investigated for the creation of models based on BERT technology for binary classification of resumes. For each of these variants, several test series with changing hyper parameter values were created to determine the best model and two cross-validation procedures were used: hold-out and leave-one-out. In the second and third variants, three additional sub-models each were created by dividing them into the three categories: school history, professional history and skills.

In addition, a fourth variant based on text similarity was chosen instead of a binary classification approach. The idea was to write a program (with the use of a BERT model) that compares the resumes submitted to a specific job posting for similarity to it (assuming that potentially suitable resumes have a higher similarity to the job posting than unsuitable ones). After analyzing for textual similarity, the final step is to output the most textually similar resumes in descending order and sort them in the file directory.

After conducting and discussing the experiments, it can be said that BERT v1, v2 and v3 have difficulties in generalizing the provided resume data, despite the use of varying hyper parameter values and optimization of the models. Based on the results of the test series, it can be stated that the recognition of unique features of a resume labeled as "acceptance" or "rejection" has been difficult.

Two key measures could improve this situation in the future:

First, with the quantitative measure: in which a larger data set consisting of several hundreds or better thousands of resumes is used. The exact quantity at which significant improvements

can be achieved is difficult to determine in advance without further research and knowledge of the data. However, it would be a good approach to first increase the amount of data by a factor of 5-10 in order to make the effects more visible.

Second, with the qualitative measure: instead of using a variety of different job profiles focus only on one or a few which are similar in the requirements to the applicants. This would improve generalization capability of the BERT models through more common and easily identifiable features.

In addition to these two measures, presumably the introduction of a predetermined format for resumes, as well as the use of another machine learning model for category recognition and parsing, could ensure efficiency as well as effectiveness in processing the data, even with increased data volumes.

In the case of the fourth variant (BERT v4), a higher data set of resumes and job advertisements as well as further test series can be used to explore in more detail whether a general use for pre-selection is better suited based on the text similarity method. In general, it can be assumed that with a significant and increasing increase in the amount of data as well as a reduction in the variation of job profiles, binary classification is more suitable, as features become clearer, contexts become more assignable and thus classification becomes easier and more accurate.

Supplemental Information

A. 1. Source code

All the programs of this projects are available at:

www.github.com/knowledgeseeek3r/Transformer-Technologies-for-analysing-applicant-data

All models are zipped as one file (3.77 GB) and available at:

www.drive.google.com/file/d/1hEU8becvVhvjyXmL_dcMkDVOSiYp8CEI/

A. 2. Code snippets

```
1  from torch.utils.data import TensorDataset ,
2      DataLoader , RandomSampler , SequentialSampler
3
4      # Convert labels to torch.Tensor
5      train_labels = torch.tensor(y_train)
6      val_labels = torch.tensor(y_val)
7
8      # Define batch size
9      batch_size = params["training"]["batch_size"]
10
11     # Create the DataLoader for our training set
12     train_data = TensorDataset(train_inputs , train_masks , train_labels)
13     train_sampler = RandomSampler(train_data)
14     train_dataloader = DataLoader(train_data , sampler=train_sampler ,
15     batch_size=batch_size)
16
17     # Create the DataLoader for our validation set
18     val_data = TensorDataset(val_inputs , val_masks , val_labels)
19     val_sampler = SequentialSampler(val_data)
20     val_dataloader = DataLoader(val_data , sampler=val_sampler ,
21     batch_size=batch_size)
```

Listing .0.1: Dataloader to create batches and tensors

A. 3. Further graphs

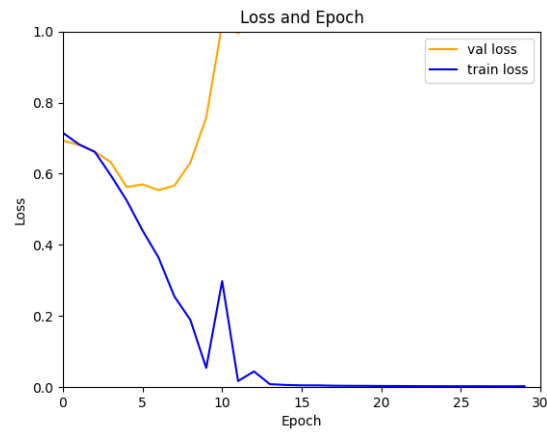


Figure .1: Epoch 1-30 for batch size 2, learning rate $5e^{-6}$ (BERT v1)

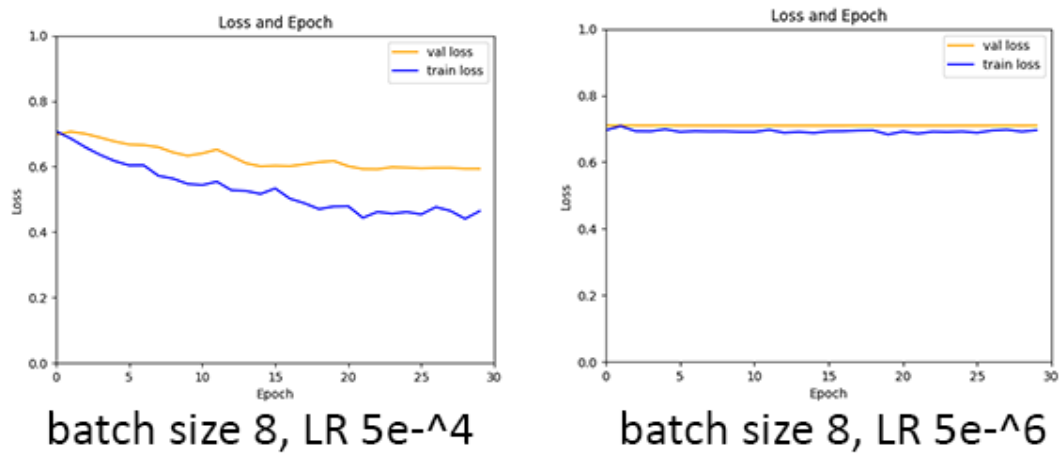


Figure .2: Epoch 1-30 for batch size 8, learning rate $LR\ 5e^{-4}$ (BERT v3 skills submodel)

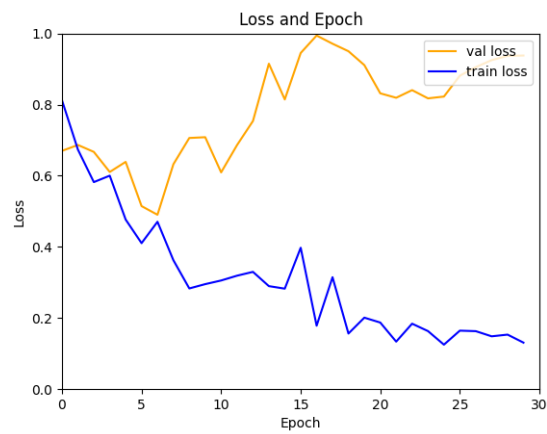


Figure .3: Epoch 1-30 for batch size 8, learning rate $LR\ 5e^{-3}$ (BERT v2 skills submodel)

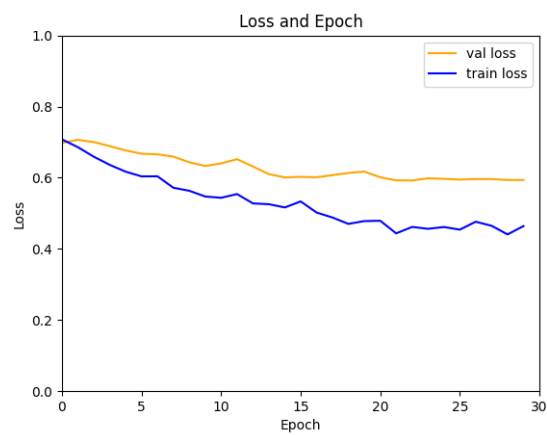


Figure .4: Epoch 1-30 for batch size 8, learning rate $LR\ 5e^{-4}$ (BERT v2 skills submodel)

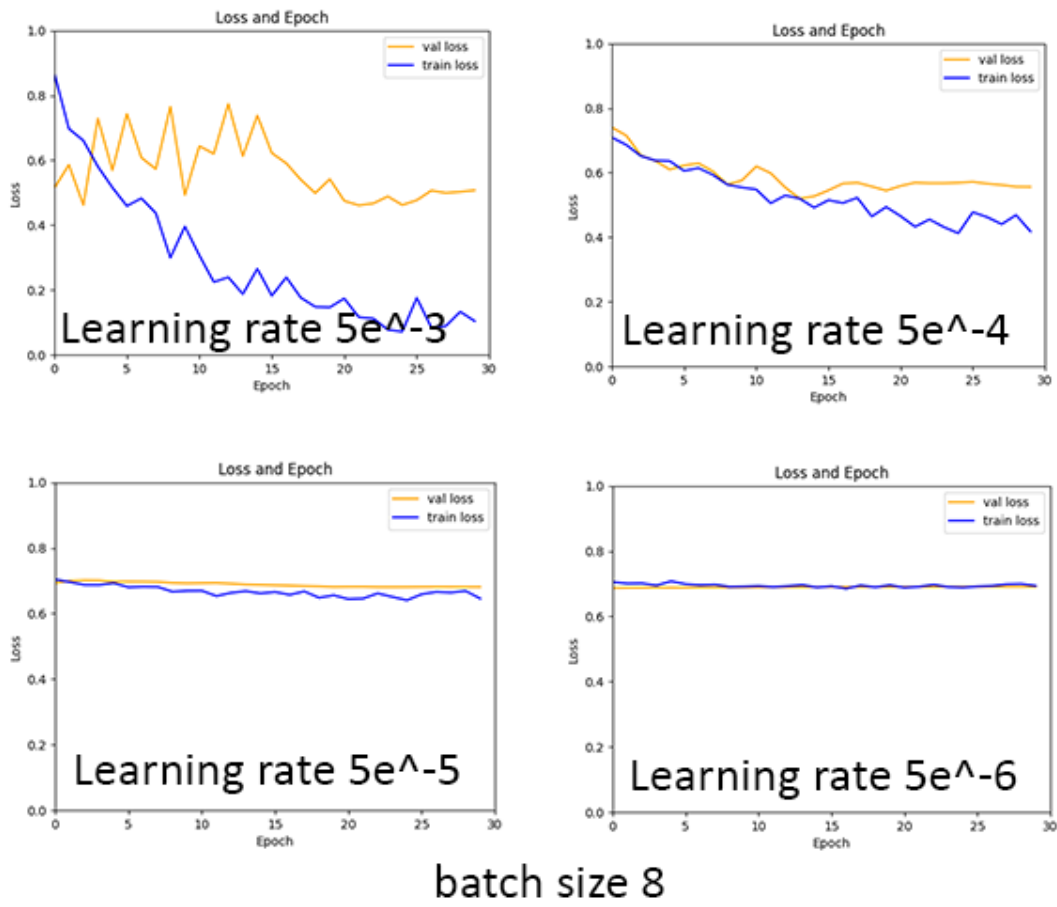


Figure .5: Epoch 1-30 for batch size 8, learning rate LR $5e^{-3}$, $5e^{-4}$, $5e^{-5}$, $5e^{-6}$ (BERT v3)

List of Figures

2.1	Word lengths of the individual resumes	9
2.2	Breakdown of data for the variants 1-3	11
2.3	data level BERT v4	12
3.1	BERT architecture [11, p. 3]	14
3.2	Corresponding vectors to tokens	15
3.3	Matrix Z of dimensions	15
3.4	Previous Matrix Z together with functions to add positions	16
3.5	Multi-Head Attention [11, p. 4]	18
3.6	Hold-out method for model evaluation [15]	19
3.7	BERT variant 1	21
3.8	BERT variant 2	23
3.9	BERT variant 3	25
3.10	BERT variant 4	26
3.11	Example of directory after running text similarity program	27
4.1	Overfitting and underfitting (All layers activated)	31
4.2	Comparison frozen layer on and off (learning rate $5e^{-4}$)	32
4.3	Best result: $5e^{-4}$, FL on, seed 40, LR Exp. off (BERT v1)	33
4.4	ROC Curve (BERT v1)	33
4.5	Best result: batch size 8, $5e^{-4}$, FL on, seed 100, LR Exp. off (BERT v2 school submodel)	35
4.6	ROC Curve (BERT v2 school submodel)	35
4.7	Best result: Batchsize 8, $5e^{-6}$, LR Exp. on, FL on, Seed 100 (BERT v2 profession submodel)	37
4.8	ROC Curve (BERT v2 profession submodel)	37
4.9	Best result: batch size 8, $5e^{-3}$, FL on, seed 100, LR Exp. off (BERT v2 skills submodel)	39
4.10	ROC Curve (BERT v2 skills submodel)	39
4.11	Best result: batch size 4, $5e^{-4}$, FL on, seed 100, LR Exp. off (BERT v3 school submodel)	41
4.12	ROC Curve (BERT v3 school submodel)	41
4.13	Best result: batch size 8, $5e^{-6}$, FL on, seed 100, LR Exp. on (BERT v3 profession submodel)	43

4.14 ROC Curve (BERT v3 profession submodel)	43
4.15 Best result: batch size 8, $5e^{-6}$, FL on, seed 10, LR Exp. off (BERT v3 skills submodel)	45
4.16 ROC Curve (BERT v3 skills submodel)	45
.1 Epoch 1-30 for batch size 2, learning rate $5e^{-6}$ (BERT v1)	60
.2 Epoch 1-30 for batch size 8, learning rate LR $5e^{-4}$ (BERT v3 skills submodel) . .	60
.3 Epoch 1-30 for batch size 8, learning rate LR $5e^{-3}$ (BERT v2 skills submodel) . .	61
.4 Epoch 1-30 for batch size 8, learning rate LR $5e^{-4}$ (BERT v2 skills submodel) . .	61
.5 Epoch 1-30 for batch size 8, learning rate LR $5e^{-3}$, $5e^{-4}$, $5e^{-5}$, $5e^{-6}$ (BERT v3) .	62

List of Tables

2.1	Raw data segmentation	4
2.2	Used resume data for BERT v4	5
4.1	Varying batch size and learning rate (BERT v1)	30
4.2	LR $5e^{-6}$ with and without exponential learning rate, frozen layers, dropout (BERT v1)	31
4.3	LR $5e^{-4}$ with and without exponential learning rate, frozen layers, dropout (BERT v1)	31
4.4	Best experiment with different seeds (BERT v1)	32
4.5	Varying batch size and learning rate (BERT v2 school submodel)	34
4.6	LR $5e^{-4}$ with and without exponential learning rate, frozen layers, dropout (BERT v2 school submodel)	34
4.7	Best experiment with different seeds (BERT v2 school submodel)	34
4.8	Varying batch size and learning rate (BERT v2 profession submodel)	36
4.9	LR $5e^{-6}$ with and without exponential learning rate, frozen layers, dropout (BERT v2 profession submodel)	36
4.10	Best experiment with different seeds (BERT v2 profession submodel)	36
4.11	Varying batch size and learning rate (BERT v2 skills submodel)	38
4.12	LR $5e^{-3}$ with and without exponential learning rate, frozen layers, dropout (BERT v2 skills submodel)	38
4.13	Best experiment with different seeds (BERT v2 skills submodel)	38
4.14	Varying batch size and learning rate (BERT v3 school submodel)	40
4.15	LR $5e^{-4}$ with and without exponential learning rate, frozen layers, dropout (BERT v3 school submodel)	40
4.16	Best experiment with different seeds (BERT v3 school submodel)	40
4.17	Varying batch size and learning rate (BERT v3 profession submodel)	42
4.18	LR $5e^{-6}$ with and without exponential learning rate, frozen layers, dropout (BERT v3 profession submodel)	42
4.19	Best experiment with different seeds (BERT v3 profession submodel)	42
4.20	Varying batch size and learning rate (BERT v3 skills submodel)	44
4.21	LR $5e^{-6}$ with and without exponential learning rate, frozen layers, dropout (BERT v3 skills submodel)	44
4.22	Best experiment with different seeds (BERT v3 skills submodel)	44

4.23	Best Results of BERT v1, v2 and v3 experiments	46
4.24	Assignment of the row number to the model name	46
4.25	Job profile: Qualitätsplaner and model: base-cased-german	47
4.26	Job profile: Qualitätsplaner and model: BERT v1	47
4.27	Job profile: Qualitätsplaner and model: multilingual	48
4.28	Job profile: Fertigungsleiter Spritzguss and model: base-cased-german	48
4.29	Job profile: Fertigungsleiter Spritzguss and model: BERT v1	49
4.30	Job profile: Fertigungsleiter Spritzguss and model: multilingual	49

List of Listings

2.3.1	Cleaning with Regular Expression	7
2.3.2	Preprocessing data of resumes	8
2.3.3	Preprocessing test and validation data	9
3.2.1	iteration for leave one out cross validation	20
3.3.1	final score function	24
.0.1	Dataloader to create batches and tensors	59

Bibliography

- [1] A. Edlich, G. Phalin, R. Jogani, and S. Kaniyar, “Driving impact at scale from automation and ai,” *Digital McKinsey*, Feb. 2019, last checked on 2021-08-19. [Online]. Available: <https://www.mckinsey.com/~media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Driving%20impact%20at%20scale%20from%20automation%20and%20AI/Driving-impact-at-scale-from-automation-and-AI.ashx>
- [2] G. Herzberg, R. Panikkar, R. Whiteman, and A. Sahu, “The imperatives for automation success,” *Digital McKinsey*, Aug. 2020, last checked on 2021-08-19. [Online]. Available: <https://www.mckinsey.com/business-functions/operations/our-insights/the-imperatives-for-automation-success>
- [3] E. Chambers, M. Foulon, H. Handfield-Jones, and S. Hankin, “The war for talent,” *McKinsey Quarterly*, p. 48, Jan. 1998, last checked on 2021-08-19. [Online]. Available: https://www.researchgate.net/publication/284689712_The_War_for_Talent
- [4] V. Bhatia, P. Rawat, A. Kumar, and R. Shah, “End-to-end resume parsing and finding candidates for a job description using bert,” *arXiv:1910.03089 [cs]*, Sep. 2019, last checked on 2021-08-19. [Online]. Available: <https://arxiv.org/pdf/1910.03089.pdf>
- [5] “Regular expression operations,” 2021-08-18, last checked on 2021-08-19. [Online]. Available: <https://docs.python.org/3/library/re.html>
- [6] S. Kotamraju, “How to use bert from the hugging face transformer library,” *Towards data science*, Jan. 2021, last checked on 2021-08-19. [Online]. Available: <https://towardsdatascience.com/how-to-use-bert-from-the-hugging-face-transformer-library-d373a22b0209>
- [7] “What is padding in a neural network?” 2020-02-07, last checked on 2021-08-19. [Online]. Available: <https://www.machinecurve.com/index.php/2020/02/07/what-is-padding-in-a-neural-network/>
- [8] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, “Don’t decay the learning rate, increase the batch size,” *arXiv:1711.00489 [cs]*, Feb. 2018, last checked on 2021-08-19. [Online]. Available: <https://arxiv.org/pdf/1711.00489.pdf>

- [9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *aclanthology:N19-1423*, Feb. 2018, last checked on 2021-08-19. [Online]. Available: <https://aclanthology.org/N19-1423.pdf>
- [10] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in bertology: What we know about how bert works," *ACL Anthology*, Feb. 2020, last checked on 2021-08-19. [Online]. Available: <https://aclanthology.org/2020.tacl-1.54.pdf>
- [11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is all you need," *acm digital library*, pp. 3,4,5, Dec. 2017, last checked on 2021-08-19. [Online]. Available: <https://dl.acm.org/doi/pdf/10.5555/3295222.3295349>
- [12] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," p. 5, Jun. 2016, last checked on 2021-08-19. [Online]. Available: <https://sebastianraschka.com/blog/2016/model-evaluation-selection-part1.html>
- [13] "Huggingface," 2019-06-14, last checked on 2021-08-19. [Online]. Available: <https://huggingface.co/bert-base-german-cased>
- [14] j. devlin, m.-w. chang, k. lee, and k. toutanova, "How to use bert from the hugging face transformer library," *Aclanthology*, p. 4, May 2019, last checked on 2021-08-13. [Online]. Available: <https://aclanthology.org/N19-1423.pdf>
- [15] A. Kumar, "Hold-out method for training machine learning models," *Vitalflux*, Dec. 2020, last checked on 2021-08-28. [Online]. Available: <https://vitalflux.com/hold-out-method-for-training-machine-learning-model/>
- [16] T.-T. Wong, "Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation," *ScienceDirect*, p. 4, 2015, last checked on 2021-08-19. [Online]. Available: <https://reader.elsevier.com/reader/sd/pii/S0031320315000989>
- [17] "python-course," last checked on 2021-08-19. [Online]. Available: <https://www.python-course.eu/softmax.php>
- [18] J. Han, M. Kamber, and J. Pei, Eds., *Data Mining: Concepts and Techniques*. Burlington, Massachusetts, U.S.: Morgan Kaufmann, 2012, last checked on 2021-08-19.
- [19] J. M. Bland and D. G. Altman, "Statistics Notes: Diagnostic tests 1: sensitivity and specificity," *The BMJ*, 1994, last checked on 2021-08-19. [Online]. Available: <https://www.bmj.com/content/308/6943/1552>