Leveraging LSTM and Language Embeddings for Age Group Estimation in Child Language Data

Marcus René Gullerud

Thesis submitted for the degree of

Master in YOUR Master's Program Name

60 credits

Department of Informatics

The Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2023

Leveraging LSTM and Language Embeddings for Age Group Estimation in Child Language Data

Marcus René Gullerud

Leveraging LSTM and Language Embeddings for Age Group Estimation in Child Language Data

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Contents

Abstract

This thesis delves into the potential of artificial intelligence (AI), particularly deep learning, in predicting the age group of children based on their conversational text, ranging from ages 3 to 15. It employs Long Short-Term Memory (LSTM) networks, a type of recurrent neural network optimized for sequence-related tasks such as text classification. The research navigates the unique challenges inherent in this task, including the distinct linguistic patterns of children, varied developmental stages, and the imbalance within the dataset. The thesis further examines the appropriate evaluation metrics for an imbalanced dataset, the significance of cross-validation, and the potential improvements for the model. While the research uncovers fascinating insights, it also identifies areas for future exploration. The results underscore the complexity of age-specific text classification, highlighting the need for a more refined understanding of children's cognitive and linguistic development and the further evolution of AI models.

Acknowledgments

I would like to extend my heartfelt appreciation to my research supervisors, Dr. Pål Halvorsen, Dr. Michael Riegler, and Syed Zohaib Hassan, for their invaluable advice, patience, constant motivation, and unwavering support throughout the process of writing this thesis. Their insightful feedback and guided mentorship have been instrumental to the completion of this work, for which I am deeply grateful.

I would also like to extend my gratitude to my family and friends, for their support and love.

Chapter 1

Introduction

1.1 Introduction

The overarching project of this thesis entails how interactive computer-based learning can provide effective and productive investigative interviews of minors[34], that are victims of sexual abuse. In recent years there have been vast improvements in the field of digital avatars, characters, and agents that are near realistic, with transfer knowledge and learning process provided by the area of Artificial Intelligence in Education (AIE). The main goal is to create a realistic child avatar, character, digital human[43], based on different components and interactions together, which includes dialogue models, and auditory, emotional, and visual components[17].

1.2 Motivation

Child age classification is a challenging task in natural language processing and artificial intelligence. The ability to accurately determine a child's age based on text can have significant implications for various applications, such as content filtering, personalized recommendations, educational tools, and most importantly, child protection measures.

Classifying text for children's age poses unique challenges compared

to age classification for adults. Children's vocabulary, language usage, and writing style differ a lot depending on their developmental stage. Additionally, children's text often contains simplified syntax, frequent use of specific words related to their interests, and distinct linguistic patterns [31].

A primary challenge in child age classification is the acquisition of labeled training data. Annotating large-scale data-sets with correct age labels for children can be time-consuming and costly. In addition, available data-sets are limited, with different representations across age groups, which in turn makes it difficult to build a robust model that can generalize well to various age ranges.

A second significant aspect is the ethical consideration with regard to digital data collection with minors. Regulations need to be followed when to ensure the responsible collection and use of data to protect children's privacy and protection [24].

The conversation of child age classification in the context of text analysis opens opportunities for multiple applications. It can aid in producing age-appropriate educational or recreational content for children online, and detect cyberbullying, or other forms of abuse. But, in order to achieve proper classification, careful consideration is required when approaching the unique challenge of children's development of vocabulary.

To create a setup for evaluation for classifying a child's age from text, there are some key points to consider. First, children's language evolves quickly as they age, and is reflected in their cognitive and emotional development as they express themselves[6]. Lastly, a wide range of ages, with diverse representatives in order to produce good generalization.

The field of artificial intelligence has improved in the later years with pre-trained classifiers from Meta, Google, Hugging Face, etc, which all yield terrific performance in certain fields, and unburdens the user's time, data,

and costly hardware. However, they are dependent on the corpus on which they are trained, and their models are generic which could prove interesting results when moving to the problem statement [3].

1.3 Problem statement

In light of the outlined motivations, this master's thesis endeavors to investigate the potential and precision of age group estimation for children aged 3 to 15, grounded in conversational text. This will involve utilizing pre-trained models derived from recognized communities and organizations.

The available datasets are collected from the talk bank "CHILDES" [9] which contains child interviews from different countries, and under different settings. However, the children in focus are the ones from England and America. These conversations will be the main source of data along with specific linguistic properties for added features, which we will get into later.

The research questions that will be addressed in this thesis are as follows:

1. Can artificial intelligence (AI), particularly deep learning algorithms, accurately classify text suitable for children within the specific age range of 3 to 15?

2. What are the key challenges and unique characteristics encountered when classifying children's age, particularly with respect to linguistic patterns, developmental stages, and imbalance in the dataset?

3. What specific evaluation metrics should be employed for an imbalanced dataset, which additional features need to be taken into account, and what role could methodologies such as cross-validation play in this process?

1.4 Limitation and scope

1.4.1 Limitation

Deep learning modules require a large volume of trainable and labeled data in order to produce accurate estimations. A limitation encountered was the available data in this field of age-specific content. The most amount of data found in the talking bank was centered around ages 4, 5, and 6, with very little trainable data on ages 7 and up. This resource limitation hampered the AI model's ability to accurately classify a child's age, especially for the older age groups. Furthermore, because of the restrictive data on children, unintended bias toward the over-represented classes can happen to achieve overall accuracy. In turn, this may also affect classification from the pre-trained classifiers when producing child age estimation as the training data needs to be complex and is limited in this field, and the availability of pre-trained modules for such tasks.

A key difference between many other classification tasks and age estimation is that the boundaries are fairly unclear, which is another limitation. The use of language develops on an individual level, therefor age can sometimes be arbitrary, and overlap with different age groups.

1.4.2 Scope

There are many different tasks in the field of text classification such as spam detection, sentiment analysis, topic labeling, language detection, and many more. The thesis work is confined to text classification of age group estimation, and the scope of the thesis is to explore and gain insight into the potential of determining the age of a child from conversation text. Furthermore, explore the potential of LSTM-based deep-learning architecture with fine-tuning for age detection, and use pre-trained transformers from Hugging Face, such as BERT and RoBERTa for generating embedding.

1.5 Research methods

There are several research methods that can be applied to a study, and more than one can be applied to one's research. Research methods are used to keep the workflow structured and steady, and are considered the building block of a thesis. They provide different utilities for information gathering, and how that information is managed.

1.5.1 Research paradigm: Methodology

The methodology research paradigm concerns itself with a theoretically informed way to approach data production. Before the work has begun, it is the strategy, course of action, process, and design that helps to decide the form of research method, and how that method is to be applied to one's research. Additionally, it is a deciding factor for which data are required for the research, and which tools to use when gathering appropriate information [37].

1.5.2 Empirical Approach

Empirical research techniques are a part of daily life, as it is a part of regular studies, and is based on direct and indirect observations. The technique in researching context involves planning why, when, and what to observe, with the intention of reducing misleading and poor interpretations. For the thesis, the empirical approach is well-suited for data gathering, design, and analysis [33].

1.5.3 Mixed Approach

A mixed approach is the most practical. A quantitative method relies on statistics and larger sample sizes to assume that your data contain good generalization and representation. The drawback is the lack of depth in the research. A qualitative method allows for lots of detail about specific cases, groups, and people. However, it disables you from making a general statement and is very time-consuming.

A mixed approach collects both of these approaches, to answer questions and to compliment each other. This project includes a variety of interview objects and categorizes them by age groups, with additional features that are unique to them [2].

1.6 Ethical considerations

1.6.1 Data collection

The data set collected for this thesis is provided by the "childes" talk bank[9] and is properly cited throughout the thesis. The conversational data sets are publicly available, and according to the project descriptions, each child and parent are recruited and willingly participating. The talk bank has little identifiable information about each child available, mostly the first name, gender, and age. The reviewed conversations are structured around book readings, free conversations between an interviewer or parents, games, and child-on-child conversations. This may capture the cognitive and linguistic development in a more general sense, and produce natural responses from the children. The key ethical consideration when gathering data based on children is the power dynamic during a conversation, which without proper facilitation may lead to inauthentic expressions [28].

1.7 Main Contributions

This thesis serves as a demonstration of artificial intelligence's capability, specifically deep learning algorithms such as LSTM networks, to predict children's age groups based on their conversational text. This approach aligns with the research question concerning the capability of AI to classify text associated with different age ranges of children.

In addition, a detailed analysis of the unique challenges and characteristics encountered when classifying children's age based on linguistic patterns, developmental stages, and dataset imbalances is provided. By outlining

novel strategies and potential solutions for overcoming these challenges, the research question about identifying and managing the key challenges in this task is addressed.

Through the identification of specific evaluation metrics suitable for handling imbalanced datasets, such as balanced accuracy, and the exploration of cross-validation as a useful methodology to increase the robustness of the model evaluation process, the study also successfully responds to the research question concerning the appropriate evaluation metrics and methodologies for this task.

Crucially, the research highlights key areas that warrant further development, such as the need for incorporating more advanced linguistic features and exploring alternative machine learning models. This not only serves as a roadmap for future research and potential improvements in the field but also enhances the study's overall contribution.

By leveraging theories of cognitive development and integrating them into the research design, the study also offers new perspectives on how these theories can be applied to AI-driven age classification. This contribution provides context for the observed results and contributes to broader discussions on the interplay between cognitive development and language use.

1.8 Thesis outline

1.8.1 Chapter 2 - Background

The background chapter will outline in detail the required knowledge necessary before moving on with further chapters. It will provide information about the relevant technology utilized, include information about the linguistic development of children, and lastly related research within the field of age detection with their respective results.

1.8.2 Chapter 3 - Methodology

This chapter outlines the methodological choices underpinning the research. It provides an extensive account of the dataset compilation and presents the step-by-step procedure undertaken to preprocess the conversation samples. Furthermore, it discusses the linguistic principles that guided the integration of additional features into the dataset. The utilized deep learning model's design, parameters, and evaluation metrics are meticulously explained.

1.8.3 Chapter 4 - Results

The result chapter will present the outcomes of the applied methodology. It offers a detailed presentation of the results obtained from the deep learning model, with interpretive commentary on notable patterns.

1.8.4 Chapter 5 - Discussion

This chapter involves an in-depth evaluation of the research findings in relation to the initial objectives. It examines the practicalities, challenges, and learning experiences derived from implementing deep neural networks for model training.

1.8.5 Chapter 6 - Conclusion

The thesis reaches its culmination in this chapter. It draws together the primary insights from the research, summarizing the extent to which the research objectives were met. This chapter concludes with a consideration of potential avenues for future research, thereby paving the way for further scholarly discourse in this field.

Chapter 2

Background

In this chapter, we provide background information for the relevant technologies used in the thesis work in order to understand and recognize the approach taken. We will begin with an introduction to some of the technical aspects of machine learning that have been used and are mentioned throughout this thesis. Then, introduce some linguistic characteristics produced by children. Lastly, we will present and discuss other relevant and related projects in the field of age detection using deep learning.

2.1 Relevant technology

2.1.1 BERT

Bert is a language representational model that stands for "Bidirectional Encoder Representation from Transformer", and is used to pre-train unlabeled text with deep bidirectional representation. BERT jointly conditions the right and left side context in all layers, resulting in the model's ability to be fine-tuned with only one extra output layer to produce a high-quality model for a variety of tasks, such as age estimation in this instance [13].

The BERT architecture consists of a stack of transformer encoder layers with many self-attention "heads". When a word or sentence is passed as an input token in a sequence, each head computes a key, value, and query vector that is used to create a corresponding weighted representation. When all the tokens are vectorized, all the output heads in the same layer are combined and run through a fully connected layer. The layers are wrapped in a skip connection and a layer normalization [11]. The workflow of BERT is divided into two parts, pre-training, and fine-tuning. Pre-training falls into the category of masked language modeling (MLM), and next sentence prediction (NSP). This involves the prediction of randomly masked input tokens, and if two inputs are close to each other in a vector space. Fine-tuning consists of one or many fully connected layers that are stacked on the final encoder layer typically [38]. The input follows a computational path through the BERT model. First, the input is tokenized by each word, then passed through the token, position, and segment layers and combined into a fixed-length vector. The special tokens [CLS] and [SEP] are respectively used for classification prediction, and to separate the input segments [38].

The embeddings created through the BERT model contain a contextual word representation of the input vectors. The BERT model is trained on an enormous corpus as a language model and is able to produce context-sensitive embeddings of word representations for each word in a sentence, which in turn can be fed into a downstream task such as age prediction with deep learning [1].

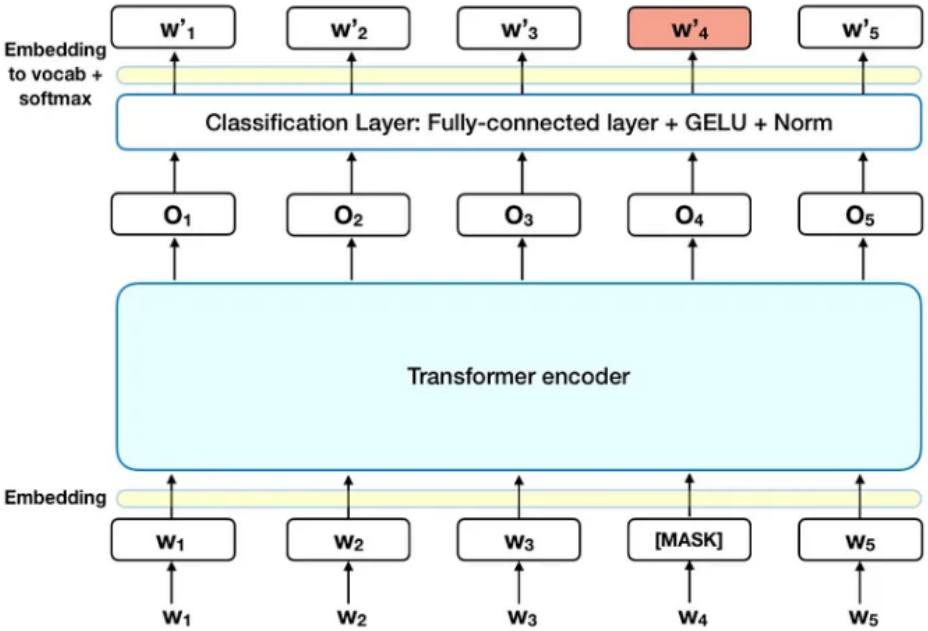

Figure 2.1: Visual representation of the BERT architecture, from towards sience[32]

2.1.2 RoBERTa

RoBERTa is an extention of the BERT model, and therefore also a transformer. It serves the same purpose as BERT, by being developed for sequence-to-sequence modeling. It is comprised of the same components such as tokenizer, transformers, and heads. To reiterate, the tokenizer transform text into sparse index encodings, and the transformer creates contextual embeddings from these encodings. Lastly, the head wraps the transformers so that the embedding can be used for the downstream task [46].

What separates RoBERTa from the classic BERT transformer is the corpora it was trained upon. It is trained on 4 different corpora, with 50 000 character level Byte-Pair Encoding, compared to BERT's 30 000. Apart from the corpus size that it is trained upon, RoBERTa uses longer sequences, and has a longer run time.

2.1.3 Natural Language Processing

Natural Language Processing (NLP) is a branch of artificial intelligence/AI, that allows computers to respond to text, speech or images in a similar way that humans do. This field combines computational linguistics (CL) with statistics, AI, and deep learning modules [19]. Some common tasks where the field of NLP is deployed involve information extraction, sentiment analysis, summarization, automatic speech, image recognition, translation, and text prediction. Computational linguistics is a sub-field of NLP that uses computational techniques to learn, categorize and understand human language. The purpose is to aid with human-to-human communication, as well as human-to-machine communication, as in Machine translation(MT). Human languages are nuanced and ambiguous; considering double meanings in words and phrases, the use of pronouns in different languages, and many more. Therefore, one of the structural approaches is to develop a module inhabiting a vast number of words in their simplified versions.

2.1.4 Neural networks

Artificial neural networks, or neural networks (NN) for short, are computational models trying to simulate the process of thinking. NN produces results through data gathering, pattern detection, relationship data, and training. NN consists of layers with a ranging number of neurons(or nodes): The input layer, hidden layer(s), and the output layer. The neurons are connected to one another layer by layer, with an associated numerical weight. There are a few attributes to take notice of when deploying a NN. The weight is a coefficient, representing the signal or strength of the connection between two neurons. A higher weight will have a larger influence on the output, making the activation function trigger earlier. Bias is another constant, acting as an intercept in a linear equation. It is used to adjust the output in the NN, helping the model in a way to best fit the data. Weights and bias work together on a neuron, and the process goes as such:

OUTPUT = SUM(WEIGHTS * INPUTS) + BIAS

2.1.5 Deep Learning Frameworks

Deep Learning Frameworks are interfaces, tools, and libraries, usually open-source. These frameworks help with the implementation of machine learning and AI and make it easier for people with little experience in the field to integrate them. These interfaces and tools provide the means to upload datasets and train modules in Deep Neural Networks. There are a few open-source frameworks developed, each built upon similar or different structures. TensorFlow was developed by Google and is an open-source framework that allows for operations on datasets. The structure is based on tensors, which are multilinear relations between objects in a vector space. When operations are performed on data in the tensors using TensorFlow, it is done so by building a stateful dataflow graph. TensorFlow offers support for languages such as C++, Swift, and Javascript, and Google's Research datasets. PyTorch is more widely used currently and is developed by Facebook. It uses a similar structure of tensors, with

supposedly easier module creation. Since Python is so vastly practiced, it found a popular following after its release. PyTorch is based on the Torch framework, which has a Lua wrapper for module construction and does computation in C. This framework was written from the ground up. PyTorch offers great memory and optimization, control of module structures, transparent model behavior, and compatibility with NumPy. Lastly, PyTorch offers CUDA compatibility for GPU acceleration [21].

2.1.6 LSTM

The Long Short-Term Memory (LSTM) model, an extension of the traditional recurrent neural network (RNN), was developed with the primary intent of preserving information over extended periods of time. It achieves this by introducing specialized structures known as memory cells, which contain self-connections that preserve the temporal state of the network, and multiplicative gates that meticulously regulate the flow of information [16].

These multiplicative gates, characterized as input, output, and forget gates, administer the inflow and outflow of activations within the network. The integration of the forget gate addressed earlier limitations of LSTM, enabling seamless handling of continuous input streams without necessitating segmentation into subsequences [16]. Essentially, the forget gate modulates the internal state of a memory cell, by selectively discarding or retaining information through a self-recurrent connection, which is a crucial aspect in managing the cell's state over time [23].

The innovative architecture of LSTM and the presence of these gates make it possible to store and retrieve information over prolonged periods, thereby substantially alleviating the pervasive problem of vanishing gradients. This ensures that the activation of a memory cell remains intact when the input gate is closed, and the preserved information can be utilized at an appropriate future stage in the sequence by opening the output gate [16].

In sum, the LSTM's ability to maintain and manipulate its internal state over time is what makes it uniquely suited to tasks that involve understanding and prediction based on sequences of data, such as long text conversations.

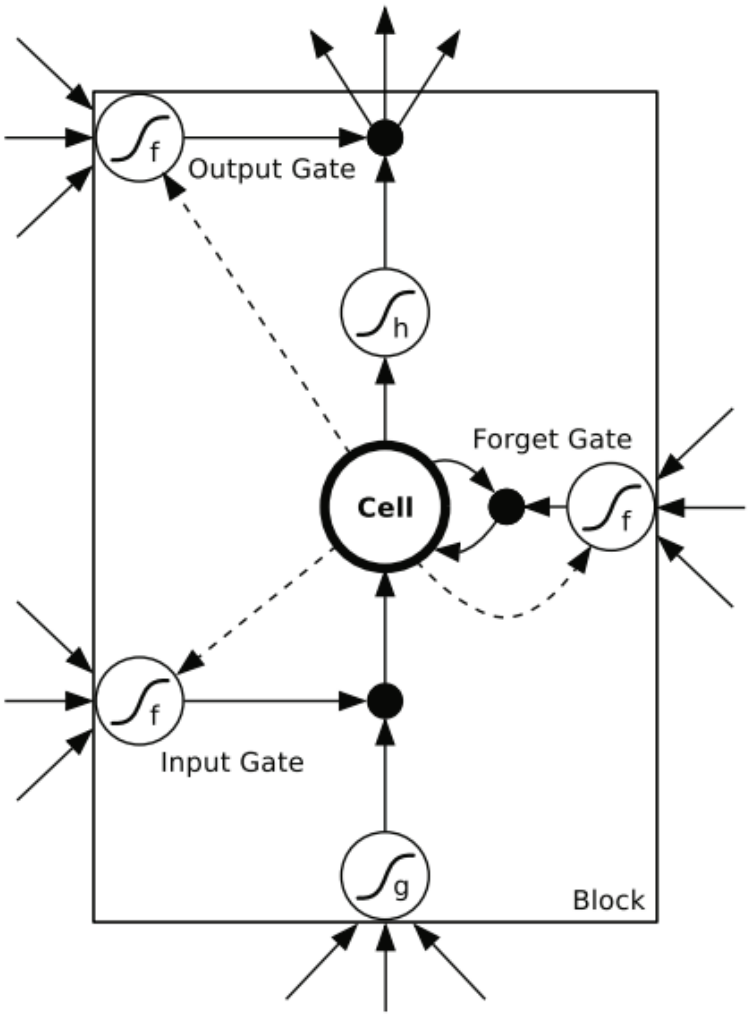

Figure 2.2: Visual representation of an LSTM block with one cell[16]

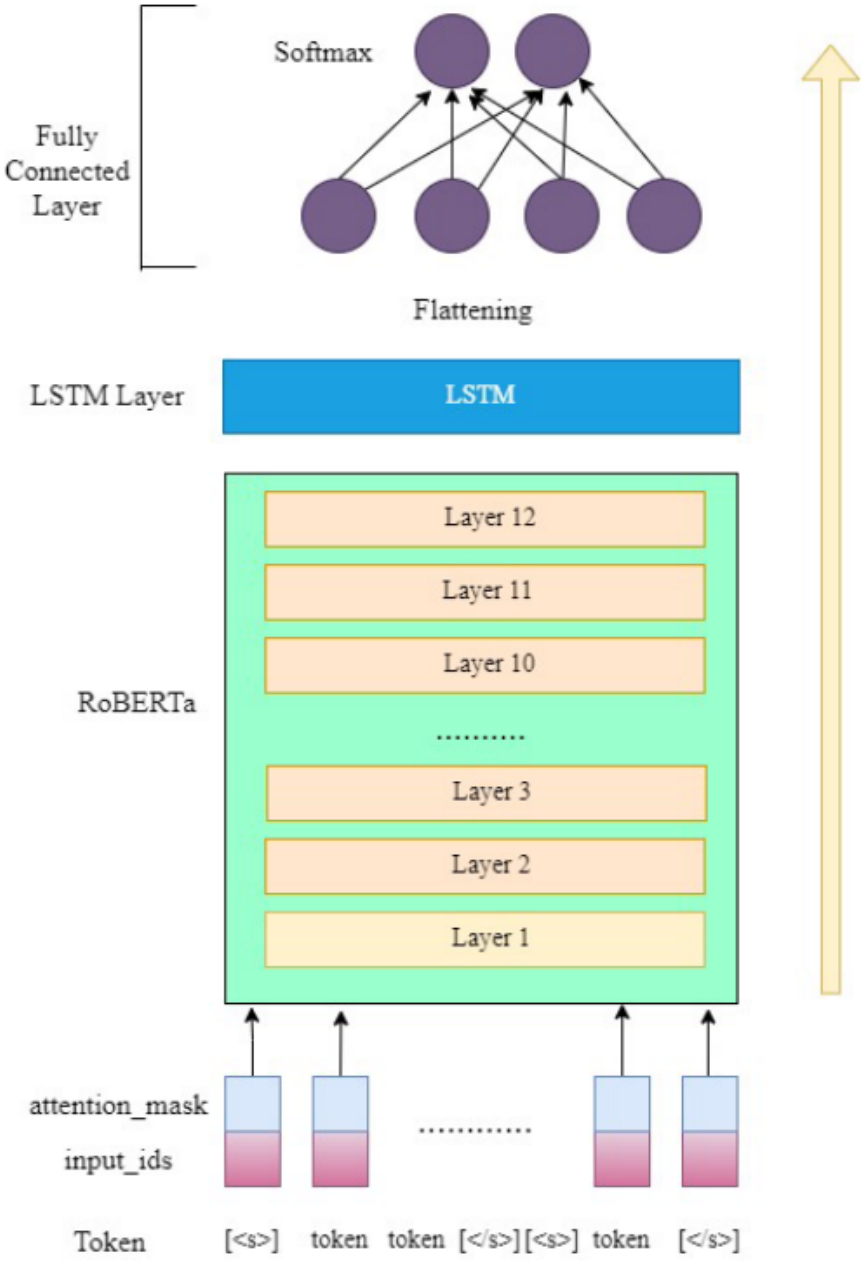

Figure 2.3: The architecture of an LSTM model with RoBERTa [46]

2.1.7 Hyperparameters

After the appropriate neural network architecture, training procedures, and regularization method are selected, the next step is to fine-tune the hyperparameters to achieve sufficient performance. This is a tedious task that demands a lot of trial and error with many different value combinations of the parameters until the researcher is left with a decent set of choices for a particular dataset. There are many different hyperparameters that can be fine-tuned, each with its specific usage and some that can auto-adjust according to the performance at run time.

Epochs are the number of how many complete passes of training on the dataset through the algorithm that has been performed. Epochs are an important hyperparameter because it determines how many times the learning algorithm will work through the dataset. It is traditionally large, so seeing numbers between 100 and 1000+ is not uncommon.

Batch size is the sample size of the dataset that is passed through the classifier. At the end of a batch, the predictions are compared to the expected output, and an error is calculated, which the algorithm uses to update the models performance. Popular batch sizes are 32, 64, and 128.

Train-test split is a procedure that divides the dataset and allows training on labeled data, as well as evaluation on unseen data. It is a common and simple process for larger data sets. The usual split performed on a dataset is 80& to train-set and 20% to test-set, but can be modified if the data is imbalanced. Libraries such as sklearn provide the option to stratify, so that each split may contain an even amount of the labeled data to each split.

Hidden layers are an essential part of the architecture of neural networks. It is positioned between the input and output layers. Their function lies in the application of weights to the inputs, which are subsequently passed through an activation function, and transforming them into outputs. Each hidden layer performs a unique nonlinear transformation of the input

data, thereby allowing the neural network to learn complex patterns and represent complex functions. Hidden layers work collectively to identify and decode the context of the input data through multiple layers of computations. They can learn abstract representations of the inputs to produce accurate and meaningful output. Therefore, hidden layers are an indispensable cog in the design of effective neural networks [12].

The Adam optimizer is an advanced optimization algorithm used in deep learning models. It is an extension of the stochastic gradient descent method that is specifically designed to handle sparse gradients on noisy problems. The strength of Adam lies in its adaptive learning rate, which makes it an efficient choice for problems involving large data or parameters. It works by maintaining a per-parameter learning rate that improves performance when dealing with sparse gradients, a common challenge with deep and recurrent neural networks. The algorithm optimizes the weights by updates to the moving average of gradients, and the average of squared gradients. These are computed based on the exponential decay of previous gradients, and their purpose is to dampen oscillations and accelerate convergence [4].

Weight decay is a robust technique in the domain of deep learning, renowned for enhancing model regularization. It operates by introducing a penalty to the loss function, subsequently driving the minimization of weights during the backpropagation phase. The primary objective of this technique is twofold: to prevent the model from overfitting and to mitigate the exploding gradient problem [35].

Weight decay is divided into two distinct types: L1 and L2 regularization. The latter is the more prevalent variant and is implemented by incorporating the squared sum of weights into an error term E, this is then scaled by a manually-adjusted hyperparameter, lambda. In contrast, L1 regularization uses the absolute sum of weights, deviating from the squared methodology adopted by L2 regularization [35].

A critical differential characteristic of L1 regularization lies in its capability to eliminate parameters by assigning zero weights. This attribute stems from the fact that penalties applied to zero weights decline more swiftly when a larger constant is deducted from a non-squared value [35].

Learning Rate (LR) is a configurable hyperparameter, which controls how fast the model is adapting to a problem. The LR number has a positive small number usually between 0.0 and 1.0. Selecting a LR number is important, as it can prevent the NN to converge on an effective solution. Too small of a number and it may take too long to converge. It can be selected by trial and error, done by calculating loss over time and adjusting accordingly. As a rule of thumb, decrease the LR once the loss stops improving [5].

Learning Rate Scheduler is a tool that adjusts the learning rate during training in accordance with a pre-defined schedule. It decays the LR by a set gamma step every epoch size. StepLR is one such scheduler from Pytorch [36].

Early stopping is a handler supported by PyTorch that stops the training process whenever there are no improvements after a certain number of events. The handler requires a patience value that assigns the number of events, a score function as an input to calculate the decrease in improvements, and a trainer engine to stop if the score function does not increase its score after the patience value is exceeded. Early stopping is a tool that decreases the model's time in unnecessary training when fine-tuning [20].

2.1.8 Evaluation

Accuracy is a fundamental metric used when evaluating most models. It quantifies the ratio of correct predictions made by the model to the total number of predictions. More formally, it is calculated as the sum of True Positives (TP), and True Negatives (TN) divided by the total number of

instances in the dataset, which includes TP, TN, False Positives (FP), and False Negatives (FN). The equation for accuracy is as follows [47]:

$$\frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{Number of correct predictions}}{\text{Number of all predictions}} = \frac{\text{Nr of correct predictions}}{\text{Size of Dataset}}$$

Balanced accuracy is used in multi-class classification when the data is imbalanced. It is the arithmetic mean of sensitivity/recall. The formula for balanced accuracy is [30]:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP + FN}}$$

See below for the recall definition.

$$\text{Balanced accuracy} = \frac{\text{Total sum of sensitivity of all classes}}{\text{Nr of classes}}$$

Precision is another important metric, evaluating the proportion of relevant instances among the instances identified by the model. It is computed as a ratio of TP to the sum of TP and FP, as shown [49]:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall, also known as Sensitivity or True Positive Rate, measures the fraction of relevant instances that have been retrieved out of the total amount of relevant instances. Its formula involves the ratio of TP to the sum of TP and FN [49]:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score is a composite metric that combines both Precision and Recall by computing their harmonic mean. This metric is particularly useful for dealing with imbalanced datasets as it considers both Precision and Recall in its calculations, this emphasizing the importance of balanced performance. A high F1-score is achieved only when both Precision and Recall are high. The formula for F1-score is given as [47]:

$$F1\ Score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Macro or weighted metrics are a great evaluation metric when the dataset is imbalanced. It is a straightforward method where the unweighted arithmetic mean of all the classes F1-scores are summarized, and divided by the total number of classes. This will result in a value that treats all classes equally regardless of their support. The formula will be as such[18]:

$$\frac{\text{The sum of all F1-Scores}}{\text{Number of classes}}$$

Cross-Entropy Loss is a logarithmic loss function extensively used in classification problems. This function compares the probability distribution of predicted class labels against the actual class labels. The true labels are usually binary in nature, 0 or 1, representing the absence or presence of a class, respectively.

The loss is computed for each instance by taking the logarithm of the predicted probability for the actual class. If the predicted probability is close to the actual value, the loss will be close to zero, indicating a good model performance. Conversely, if the predicted probability diverges from the actual value, the loss grows larger, signaling a poor model performance.

In essence, Cross-Entropy Loss quantifies the dissimilarity between the predicted probability distribution and the true distribution, and the model aims to minimize this discrepancy during the training process. Hence, a smaller Cross-Entropy Loss score signifies a model that is performing better, producing predictions that are more aligned with the true values [48].

Here is the formula for Cross-Entropy Loss for binary classification:

$$\text{Cross-Entropy Loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$$

Note: N is the number of instances, y_i represents the actual class label, and $\hat{y}_i$ represents the predicted probabilities. Additionally, the relevant term of

the sum is used in each case, given that y_i is either 1 or 0.

Cross-validation with Kfold is a resampling procedure that partitions the training dataset into a specified number k of equally sized subsets or 'folds'. The model then undergoes k rounds of training and validation. In each round, the model is trained on k-1 folds, using them as the training data, and the results are validated on a held-out test set.

The overall performance of the model is determined by averaging the performance measures from all the k iterations. This process helps in obtaining a less biased or less optimistic estimate of the model performance as it effectively utilizes all the available data for both training and validation.

Although K-Fold Cross-Validation is computationally intensive due to multiple rounds of training and testing, it is highly regarded for its efficient use of data. Each data point gets to be in a validation set exactly once and gets to be in a training set k-1 times. This method thus offers a comprehensive insight into the model's performance and robustness.[41].

2.1.9 CUDA

Deep Learning using Graphical processing units. Cuda is a platform for computing on one or more graphical processing units (GPU). It was developed by NVIDIA and was developed to increase the computing power by GPU acceleration. This solution has multiple use cases e.g. deep learning, machine learning, and data analytics. The GPU is widely used as an accelerator for deep learning (DL), because of its programmable architecture and availability. Training a neural network requires a long time, and doing so on a simple central processing unit (CPU) is doable, but inefficient compared to doing so on a GPU [27].

The resource-intensive process of training a module is shortened by the GPU by the ability to parallelize the training tasks with a larger amount of parameters, performing operations simultaneously. A solution like CUDA

also allows for a distribution of tasks over a cluster of GPUs if necessary. This process saves time for the development team, works faster than non-specialized hardware, frees the CPU, and prevents bottlenecks [27].

2.2 Child cognitive development

Piaget's cognitive development theory is based on four stages of general cognitive development between birth and adulthood[6]. There are a few theories about how children develop, such as Vygotsky's sociocultural study[25], Freud's psychoanalytic theory[15], and Erikson's psychosocial theory of development[10]. Piaget and Erikson's theories are similar in that children develop through interaction with the external environment, and that development occurs in the listed stages.

- **Sensorimotor (0 to 2 years)** is the first phase of children's cognitive development where the child first learns about their environment through senses and reflexes, and later becomes more abstract in terms of expressions.

- **Preoperational (2 to 7 years)** is the second phase where abstract expression becomes more prominent. The child is able to converse about past events and people not currently present. However, in this stage, the child tends to not perceive others' points of view.

- **Concrete operational (7 to 11 years)** is the third stage where problem-solving and categorization are more developed. Numerical and spatial abilities are also improved greatly at this stage.

- **Formal operational (11 through adulthood)** is the final phase where abstract thoughts and concepts are better perceived, not limited to time and individuals. Application of reasoning is more prominent, and the ability to test hypothesis and draw conclusions are formed.

2.3 Related work

Authorship attribution in more mature demographic groups has garnered increased attention in recent years, propelled by recent advancements in machine learning. Yet, the exploration of age detection in younger populations through text analysis, leveraging deep learning techniques, remains an under-researched domain. Empirical evidence substantiates this claim when surveying the scholarly literature online. Therefore, in this subsection, we will look into existing research that intersects the fields of age detection and text classification.

The table below displays the achieved performance accuracy for each respective research and publishing year. The first study looks into an Arabic demographic on Twitter to perform author detection. Author detection is the task of age, gender, and dialect detection. The study used two different variations of the LSTM model. The first was an LSTM model with three activation layers, a cross-entropy loss function, and an adam-optimizer. The second was similar but with hand-crafted features that were appended to the output layer [45]. The second study implemented a different method that included FastText, VGG-Face model, and CRNN. These tools were combined to use both text classification and image detection for author age detection on Russian social media. FastText is a tool created by Facebook for word representation and text classification [51], much like BERT. The VGG-Face model [14] was used to determine age based on images from social media profile pictures. CRNN model was represented by 1, 3, and 5 convolutional filter dimensions, and the recurrent architecture was represented by LSTM [39].

Table 2.1: Display of each study's respective accuracy of age detection

Research study	**Year**	**Model**	**Accuracy**
[45]	2019	LSTM	22.22%
[39]	2020	CRNN	65%

Existing research in the domain of age detection has demonstrated limited effectiveness in discerning age within smaller age brackets. The prevailing approach in most related studies is to allocate age into larger groups (e.g., [0-10, 10-20, 20-30, ...]) when using text as the sole determinant of age. Some researchers have attempted to enhance the precision of age estimation within narrower age ranges by employing a variety of techniques. However, there remains a clear need for more comprehensive and focused investigation in this field, extending beyond just minor age groups to encompass adults, and thereby advancing the overall understanding of age detection through text classification methods.

2.4 Summary

This chapter has offered a comprehensive exploration of the pertinent technical knowledge, previous research in the domain of age detection, and an introductory discussion on child development. The segment dedicated to relevant technology delved into the specifics of data preprocessing, highlighted the mechanics of the LSTM model, and shed light on its suitability for our research task. The section also elaborated on the concept of model fine-tuning and presented the metrics employed for evaluation, thereby equipping readers with a robust foundation for subsequent chapters.

Moreover, this chapter discussed two distinctive studies in author age detection, providing insight into the varying methodologies applied in the field. The first study innovatively employed a modified LSTM model, while the second merged image and text classification using a Convolutional Recurrent Neural Network (CRNN) to attain improved precision. The reviews of these studies underscore the inherent challenges of age detection, regardless of the age group, and illuminate the vast scope of intriguing prospects in this area of research.

Chapter 3

Methodology

The methodology helps to detail the approach, methods, and techniques used when designing a research study. This chapter will explain the methods used for data collection and different strategies for preprocessing the necessary data for the experiment. In addition, it will describe the architecture around the LSTM neural network, hyperparameter, support functions, training, validation, and testing. The main goal of the experiment is to be able to predict the age of children between the age of three and twelve based on conversational interviews with neural networks.

3.1 Datasets

This section will contain information about the dataset, which will be used for age detection. It will present where it originates, how it was collected, and how it was preprocessed for further use. Furthermore, it will detail the inclusion of additional features from the MRC Psycholinguistic Database.

3.2 Childes

The CHILDES corpora is a large organized talk bank, centered around interviews of children from age 0 to 15. The website offers such

conversations in numerous languages such as Chinese, French, Japanese, Scandinavian, English, etc. This thesis will be centered around English-speaking children to improve the interoperability and consistency in the model.

The conversations from CHILDES are for the most part based on playful subjects such as book readings, games, and guessing animals, as well as free conversations, to provoke spontaneous answers within the constraint of the conversations (Fletcher, 2004). In total, there are 8722 different conversations, while the total of usable conversations was 4826. The reason for such a drop in usable conversations was either a lack of age listed, which would not provide a label for further training, or ineligible and non-conversational words and sentences such as a description of activities and onomatopoetic.

Table 3.1: A generic representation of the interview participants with relevant information

Participant	**Role**	**Name**	**Language**	**Age**	**Sex**
CHI	Target Child	First name	Eng	Age Date	Male/Female
MOT/FAT	Mother/Father	-	Eng	-	Male/Female
INV	Investigator	First name	Eng	-	-

The notable values from the table are the tag for children and age. The format is consistent for all conversations, with “CHI” as the child and age in the format of “age;date of birth”.

List of all the corpus names from North America [7] :

- Bates
- BernsteinRatner

- Bliss
- Bloom
- Bohannon
- Braunwald
- Brent
- Brown
- Clark
- Demetras - Trevor
- Demetras - Working
- EllisWeismer
- Evans
- Feldman
- Garvey
- Gathercole
- Gelman
- Gillam
- Gleason
- Haggerty
- Hall
- Higginson
- HSLLD
- Kuczaj

- MacWhinney
- McCune
- McMillan
- Morisset
- Nadig
- Nelson
- New England
- NewmanRatner
- Nicholas
- Nippold
- Peters/Wilson
- POLER-Controls
- Post
- Rollins
- Rondal
- Sachs
- Sawyer
- Snow
- Soderstrom
- Sprott
- Suppes
- Tardif

- Valian
- Van Houten
- Van Kleeck
- Warren
- Weist

List of all the corpus names from the United Kingdom[8]:

- Belfast
- Conti-Ramsden
- Conti-Ramsden
- Cruttenden
- Edinburgh
- Fletcher
- Forrester
- Gathercole/Burns
- Howe
- KellyQuigley
- Korman
- Lara
- Manchester
- MPI-EVA-Manchester
- Nuffield

- OdiaMAIN
- QuigleyMcNally
- Sekali
- Smith
- Thomas
- Tommerdahl
- Wells

3.3 MRC Psycholinguistic database

To understand the contents of this database, one would need to understand the concept of psycholinguistics. Psycholinguistics investigates the psychological process of understanding language, and what determines which age an individual comprehends and produces language. At the core of linguistics, you will find acquisition, comprehension, and production [50]

With that, in the MRC database, there is a wide range of possible selections to choose from for database production. The relevant fields of selection are "Kucera-Francis written freq.", "Age of acquisition rating", and "Words". The second optional property allows for limitations, where the age of acquisition rating is limited by age. While this tool provides very good limitational abilities and a variety of psycholinguistic properties to narrow the results down to satisfactory results, there was, unfortunately, a limitation met. The database is truncated at 5000 lines, which therefore resulted in a search for the full MRC database from different sources, as the full database from the MRC database had been discontinued.

The full database was extracted from a public repository from GitHub, that used SQLAlchemy to communicate between Python scripts and the

database, which resulted in the retrieval of 150837 values of words and their related age of acquisition score, and Kucera-Francis written frequency.

The age of acquisition score is a point system where words are assigned a value based on when a specific word is acquired [22]. There are many different ways to retrieve the age of acquisition scores. Some researchers have acquired the scores by having adult participants estimate when they learned specific words, but the validity of these scores may be inaccurate because of factors such as frequency of use, familiarity, and estimation accuracy.

Other researchers used indices of word frequency from children's exams to study the frequency of words used in different age groups. In a study made to confirm the validity of Age of acquisition scores, and derive an objective measure, children were tasked to provide the name of pictures. Children of age 2.5 to 10.9 years named 297 pictures in total. The resulting data was matched with the adult ratings and provided evidence that adult scores are reliable and valid measures of Age of acquisition scores [29]).

The Kucera-Francis Frequency was derived from a large corpus containing one million words, and the word frequency was determined by how many times a word occurred per one million words. The corpus itself was a collection of 500 continuous samples, which had approximately 2000 words respectively [40]. There are several weaknesses in this approach. First, Kucera-Francis frequency was developed in 1967, which results in inaccurate frequency scores in terms of common parts of speech, as languages develop over time. Second, the corpus consisted of appropriate literature at the time, which again may skew the word frequency in terms of today's common part of speech. Another raised issue is that the literature incorporated was subject to publication, so the count of unusual and common words might be skewed [40].

3.4 Data retrieval

To be able to train, validate and test a model, a data set needs to be collected and properly treated in order to fit the model's needs. For this thesis, the relevant information from the mentioned CHILDES talk bank was the target age and answers from the numerous conversations. The conversational retrieval process was performed by acquiring the files in their original format from the CHILDES talk bank and taken through 3 steps

Each file was processed by a bash script to alter the extension from ".cha" to ".txt", for readability, as ".cha" extensions require a certain program to open, and was not compatible with my IDE The newly formed ".txt" files were input to a Python script that extracted the age of the child, along with the replies from the child in the conversation on a one-sentence-per-line basis. This Python script ran through a loop in another bash script on all files in the gathered data folder. Lastly, the extracted conversations were cleaned of non-Ascii characters through a RegEx filter method. The output format was two files, one named "*Filename_preprocessed.txt*" and the other named "*Filename_preprocessedNoAge.txt*" to distinguish the files from one another, as one contained the age of the interview object and the other did not.

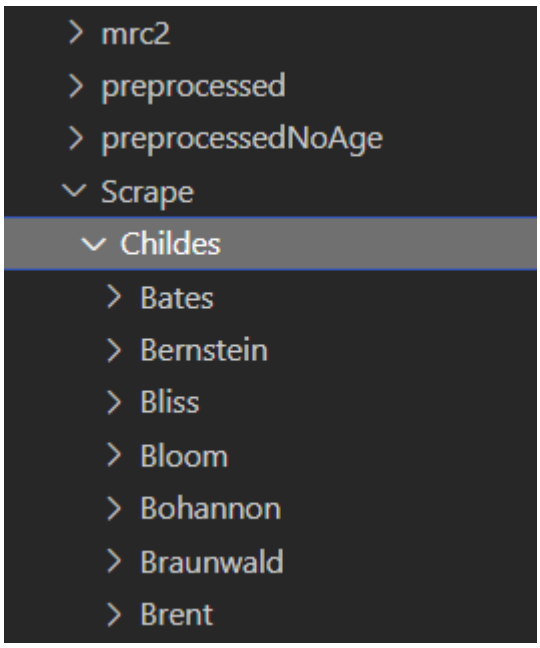

Figure 3.1: The folder structure of original and processed conversations

3.5 BERT Embeddings

The newly formed and cleaned text file without age was then transformed into an embedding. The method used was a pre-trained BERT transformer because it offers advantages such as transfer learning. A masked language model such as BERT is pre-trained on a very large corpora, and contains millions of model parameters, thus relieving the need for self-made features [26]. However, for this task, the additional features "Age of acquisition" and "Kucera-Francis frequency" were added to capture relevant psycho-linguistic features.

The added features were taken from GitHub, as the database from the official website only produced a truncated part of itself. Link to the GitHub repository: https://github.com/samzhang111/mrc-psycholinguistics. Requirements as per the repository:

- SQLAlchemy(0.9.3)
- Python 2.7

With SQLAlchemy, all the values in the MRC database were already loaded to a variable when executed. To extract the preferred values, a session query for all instances of "words", "aoa" (Age of acquisition) and "kf_freq" (Kucer-Francis frequency) in the database was done, with the exception of all non or zero values per word. The newly extracted values were stored in a text document line by line to be utilized for further pre-processing.

The process of creating embeddings was performed through three main functions: "load_kf_aoa_values", "get_embeddings", and "main". The first step was to initialize the BERT model and Tokenizer, specifically the "bert-base-uncased" model, which is trained on lowercase English text. The python script then performs a device selection to check if a CUDA-capable GPU is available for accelerated computation. The main function can be viewed in the appendix. 6.1

For each file, the sentence are loaded and tokenized into individual words or subwords using the BERT tokenizer. If a word is not contained in BERT's vocabulary, it is broken down into known subwords. Sentences that are empty after tokenization are discarded.

Next, the functions determines the structure of the BERT embeddings. It finds the maximum sentence length among all the tokenized sentences and sets up a numpy array, "file_embedding", with the dimensions of 1 x max_sentence_length x (BERT embedding size + 2). The "+2" accounts for the additional features "age of acquisition" score and "Kucera-Francis frequency", which is then stored alongside the embeddings.

The BERT embedding computation was performed on each non-empty tokenized sentence. The function "get_embeddings" convert each token into its corresponding BERT-id and forms a tensor from these ID's. This tensor is then passed to the BERT model to generate embeddings.

The function then iterates over each token in each sentence. If a token exists in the function "kf_aoa_values" Dictionary (which maps words to their corresponding age of acquisition score and Kucera-Francis frequency), then those values were stored with the embedding in "file_embedding". In cases where the tokens did not match any words in "kf_aoa_values", then only the BERT embedding was stored.

Finally, the function returns the "file_embedding" array, which now contains the BERT embeddings for each word in each sentence, along with aoa and kf_freq values where available.

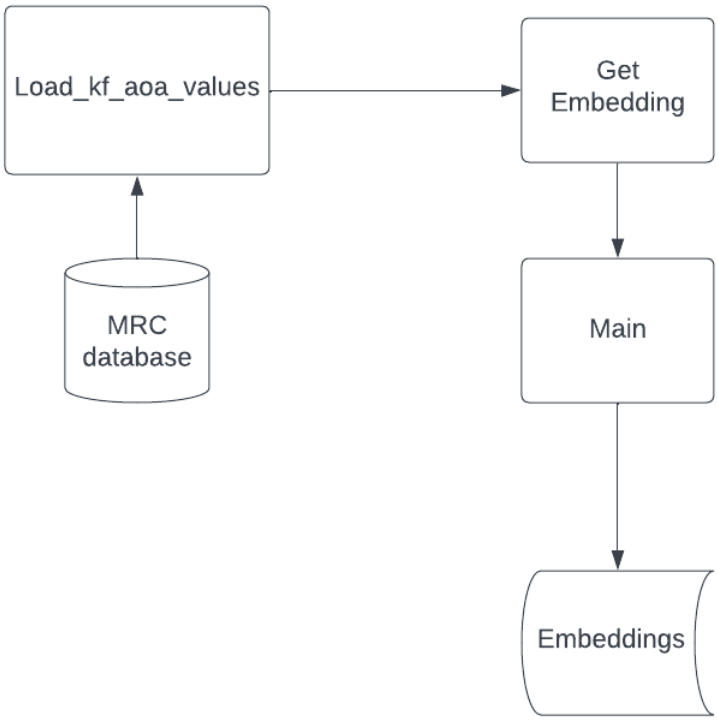

Figure 3.2: A very simple example of the embedding creation process

3.6 Data augmentation

3.6.1 Easy data augmentation

To be able to train a neural network, and have it perform well, requires a large set of data. The BERT embeddings created were sourced from approximately 4800 conversations, which yielded the same amount of embeddings. The issue was that the median age was severely disproportionate.

Table 3.2: Displays each unique age, with their respective count of conversations

Age	Count
3	739
4	880
5	605
6	254
7	373
8	148
9	174
10	115
11	66
12	7
13	42
14	33
15	35

The script contains a few helper functions in order to perform simple data augmentation.

- read_file
- random_lines_swap
- random_lines_deletion

- write_file

The list augmentations were selected to prevent unnecessary contextual loss when performed, hence the method of re-organizing the words within each sentence was not used. The main function takes in four arguments: input_folder, output_folder1, output_folder2, and ages_to_process. The inputs to the main function allow dynamic data augmentation on different sets before the augmented data is turned into augmented embeddings. The functions perform a series of operations. Creates a list of all files with "_processed.txt" in its name from the input folder, and reads the content of each file. Next, it skips empty lines or files that do not start with a comma and proceeds to the next file. The script retrieves the age contained in each separate file, and if the age does not equal the input age from "ages_to_process" it skips to the next file. After the criteria of age and non-empty files are met, it performs the listed augmentations.

The random_lines_swap functions perform file-length/2 number of swaps, of the sentences within a file. Random_line_deletion deletes random lines within a file with a probability of 0.25 per sentence.

Lastly, the script produces 2 separate files as output from each input, one that contains only sentences, and one that contains the age as well as sentences.

To summarize, the script reads files from a folder and performs random swaps and deletions of random sentences within a file. The conditions are that the sentences or the file cannot be empty, and swaps of lines are limited so that the process does not go on unnecessarily long. The probability of deletion is set to a reasonable level so that entire documents are not removed. After augmentations are complete, the new files are stored with a different naming convention so they can be held separately.

3.6.2 Back-translation

Back-translation is the process of translating text from one language to another, and then back to the original language. It is a well-established form of data augmentation that has seen increased interest in recent times with the availability and improvements of machine translation [44]. In this instance, the files produced by the 12-year-old subjects were back-translated from English to French, and back to English. As stated in the paragraph above, unnecessary contextual loss within each sentence was the goal, which back-translation could hinder to a good extent.

The script utilized the MarianMTModel and MarianTokenizer from the transformers library, OS to interact with the operating system, glob for file operations, and torch for PyTorch. The script defined several helper functions:

- write_file: Writes a list of lines to a file.
- back_translate:
- back_translate_file:
- back_translate_files

back_translate is the function responsible for the core back translation process, which involves the translation of batches of text from English to French, and then back to English. It receives the source text, tokenizers, models, and batch size as input. The function first checks for a CUDA device that is available for accelerated computation, and then processes the text in batches. The function returns back-translated English text. Back_translate_file read lines from specific files and performs back-translation with the "back_translate" function.

Back_translate_files takes an input folder, an output folder, and a fixed batch size as input parameters. It loads the MarianTokenizer and MarianMTModel and iterates over the provided list of files, and performs

back-translation per line with the "back_translate_file" function. Back translation is time-consuming, therefore the function utilizes the GPU if available, or the CPU if not. In addition, a batch size of 32 was implemented to process more data at a time to further reduce the processing time. The files are then saved to the output folder with the write_file function. Lastly, the function deletes the translation model and empties the GPU cache to free memory.

In summary, this script reads files from an input directory, performs back-translation on a text file, and writes the back-translated text back into a file in a new directory. The translations are performed in batches for efficiency and make use of GPU acceleration if available. The back translation was performed on conversations where age 12 was found, after the easy data augmentation took place, resulting in 14 new augmented conversations with a new total of 28 conversations in that specific age group. This technique was computationally heavy, hence the reason why it was only performed on the lowest represented age in the dataset.

3.7 Long short-term memory - LSTM

3.7.1 Tools

The model creation can become quite cluttered with different functions when it is set up, therefor a secondary script that can hold most of the helper functions that help fit the embeddings into the neural network is beneficial in terms of organization of the code, as well as debugging. It is easier to test functions one by one before the embeddings are passed on to the neural network.

The tool script primarily works with embeddings stored in a numpy format and pre-processed files in text format. It extracts information from these files, adjusts the size of the embeddings, groups the data according to their respective age, and loads a tuple of embedding and age.

The defined helper functions are as such:

- extract_first_number(file_name)
- extract_embedding_name(embedding_path)
- pad_truncate_sequence(seq, max_length_dim1, max_length_dim2)
- get_age_group(age, age_groups)
- load_embedding(data_dir, age_groups)
- move_embeddings(src_folder, dest_folder, age_list)

Firstly, the function "extract_embedding_name" takes in a file path to the embeddings and extracts each embeddings base name, and removes the added "NoAge_embedding" substring. This is used to return the original base name from the embeddings.

The function "extract_first_number" then takes in the base name as a file name, and constructs the file path for each embedding's base name, back to the original pre-processed file. As a reminder, when the interviews were pre-processed each file had its respective file-ID or base name and was saved with or without the age, with answers from the CHILDES conversations.

Now that each embedding could be connected to its original file with age contained using the new file path. The function reads the first line of the file and split it by commas to extract the first element, which is the age of the interviewee, and returns the value found. Should any error (FileNotFoundError, ValueError, IndexError) occurs, then the age is returned as none.

"pad_truncate_sequence" is designed to ensure that all sequences (in this case, embeddings) have the same shape, which is often a requirement for machine learning models. The function receives a sequence and two maximum length dimensions as input, which are the desired lengths of the

first and second dimensions of the sequence.

First, it checks whether the first input dimension is greater than the set max length of the first dimension. If it is, the function truncates the sequence to the max length of the first dimension, and only the elements that fit within the max length are kept. However, if the sequence from the input is smaller than the max length, it is padded with zeroes until it reaches the max length. The same process of truncation or padding is applied to the second dimension of the input sequence until the sequence reaches the set max length of dimension two.

"get_age_group" is a function that takes an age and a dictionary, and maps age groups to age ranges. It returns the age group that the provided age falls into. Should the age not fall into a group, it returns none. This function allows the creation of dynamic age range testing when the embeddings are moved into the neural network.

"load_embeddings" loads the embedding from the previously created numpy files in the provided directory, and proceeds to group them by age according to the age_group dictionary.

The function iterates over all files in the directory. For each numpy file, it extracts the original name, extracts the age with "extract_first_number", and checks whether the age falls into one of the provided age groups using "get_age_group". If the age is valid and falls into a group, then the function adds the file path to "embedding_filepaths".

Next, the function determines a suitable size for the embeddings. It loads each embedding from the file in "embedding_filepaths", records its size, and computes the 95th percentile of the sizes for both dimensions. This was done, so as to not include any outliers that were extremely large compared to the other embeddings. These percentiles are used as maximum lengths for padding and truncating the embeddings.

The function then iterates over "embedding_filepaths" again. For each file,

it loads the embedding, checks whether its first dimension is less than or equal to the maximum length determined in the previous step, and if it is, pads or truncates the embedding to have the same lengths as determined before. It also retrieves the age and age_group again.

Lastly, it appends a tuple that contains the embeddings, now all with the same size, along with the corresponding age group to the "data" list, which is then returned.

The last function "move_embedding" was created as a way to move certain ages within the data set. The function moves all numpy files from the source folder to a new one if the age, extracted from the file name (with "extract_embedding_name" and "extract_first_number"), is in the provided age list. This function was only utilized once, the ages zero, one, and two provided little to no text that embeddings could be created. This was done later in the development process.

This script is meant to be the retrieval method that is used further in the model. It links each embedding to the original conversation it was created from in order to retrieve the corresponding age and make sure that they all meet the same-size criteria that the model requires. Additionally, it helps to organize the next model script.

3.7.2 LSTM

The model utilized various libraries such as "numpy", "torch", "sklearn", and "imblearn" among others to properly load and process the data, define and train models, evaluate performance, and handle class imbalance. Additionally, it checks and uses CUDA for GPU-accelerated computation if a device was available, if not then CPU was the default device chosen.

The function "load_embedding" from the tools script was used to load the precomputed embeddings from a specified directory. The embeddings represent text data and were associated with an age group that was used for labeling. The data was then split into training, validation, and test sets.

Next, two different techniques to handle class imbalance were implemented. The first was random over/under-sampling. This technique requires a set number of desired samples from each age group, that was entered manually. Since the class imbalance was severe, the approach chosen was to reduce class zero to 500 and increase classes one and two to 500. This way, classes one and two could get more usage from the augmented files than the overly represented class zero could.

The second method was "SMOTE" from the "imblearn" library. This technique oversamples the underrepresented classes to ensure the model gets to see enough samples from all the classes present.

The specific LSTM model "AgeLSTM" is defined with a "forward" method that specifies how input data should be processed to produce an output. Here, it uses an LSTM layer followed by a dropout layer and a fully connected layer to make predictions. The dropout layer is used for regularization to prevent overfitting, and is a hyperparameter that can be tuned in order to improve the model.

The "AgeDataset" class is a custom PyTorch "Dataset" made to handle the specific format of the input data. It overrides the "__len__" and "__getitem__" methods that return the size of the dataset, and is made to return the embedding with the corresponding label instead.

The "train_and_evaluate_model" is where the model training, evaluation, hyperparameter tuning, and early stopping are all managed. The function takes in the learning rate, the hidden size of the LSTM layer, the number of LSTM layers, the size of the input data, and the number of classes. All these parameters are used to configure and train the LSTM model.

The function starts by implementing the "KFold" class from "sklearn.model_selection" module. Five k-folds were used for cross-validation, which is a technique to assess how well the model will generalize to unseen data by dividing the data into k-number of folds. Each fold is used for training "k-1" number of

times, and evaluated on the remaining fold.

For each fold, the function uses the train and validation data, and creates an instance of the "AgeLSTM" model, the loss function (Cross-Entropy Loss), and the optimizer (Adam). It also creates one of two learning rate schedulers (StepLR or ReduceLROnPlateau from the "torch.optim.lr_scheduler" library). Each of them adjusts the learning rate after a number of epochs.

The model is trained over a specified number of epochs. In each epoch, the model parameters are updated to reduce the loss on the training data. The parameters are then adjusted in accordance with respect to the gradient computed from the model parameters.

The model is evaluated on the validation data, after each epoch. If the validation loss does not decrease after a specified number of epochs (patience), then the training is stopped early. This is known as "early stopping" and is a technique to reduce overfitting. The lowest validation loss achieved is then saved.

After each fold, the function adds the computed validation loss and accuracy for all the folds and calculates the average validation loss and accuracy.

Finally, the function loops over a set of hyperparameters as part of a grid search. The hyperparameters in the search are learning rate, hidden size, and number of layers. The goal of the grid search is to determine what hyperparameters give the best performance on the validation set. In addition, it also produces a trending scale that is used for further fine-tuning.

After the training, the best model based on validation loss is evaluated on the held-out test set to give an unbiased estimate of its performance. A confusion matrix and a classification report (with precision, recall, and f1-score) were used to evaluate the performance of the model on the test set. Lastly, the script saves the state dictionary of the best model to disk and

demonstrates how to load this state dictionary into a new model instance.

3.7.3 Model training

As mentioned, the main dataset consists of unique conversations that were run through the BERT and RoBERTa transformers to create the embeddings for the LSTM model. The embeddings were labeled with an age group that the original conversation was based upon. There are 4 groups, or classes utilized, each with an age range designed to even out the datasets imbalance.

- Class 0 = [3]
- Class 1 = [4]
- Class 2 = [5, 7]
- Class 3 = [8, 15]

The model was divided into three sets: training, validation, and test set, with a stratified split percentage of 70/30. This resulted in a split that provided evenly split data per age group to each set, and enough data for training, validation, and testing. After the split, the augmented data was appended to the list of training data, so as not to contaminate the validation and test set. Lastly, the embeddings now with the proper label were fed into the classifier. The tabels below present the model and both of the best hyperparameter combinations for each setup that was found when fine-tuning.

Model:

Algorithm	RNN
Decoder Formation	LSTM, BERT, RoBERTa

Table 3.3

Setup with BERT-uncased:

Hyperparameters	Values
Learning rate	0.009
Hidden size	64
Number of layers	3
Dropout	0.4
Epochs	300
Batch size	32
Patience	15

Table 3.4: Best state model results with BERT

Setup with RoBERTa:

Hyperparameters	Values
Learning rate	0.009
Hidden size	64
Number of layers	2
Dropout	0.4
Epochs	300
Batch size	32
Patience	15

Table 3.5: Best state model results with RoBERTa

3.8 Summary

This chapter delves into the usage and creation of the CHILDES corpora, supplemented by features from the MRC Psycholinguistic Database. It details the process of embedding creation leveraging transfer learning benefits, data augmentation techniques, and handling class imbalance. Furthermore, it outlines the development, training, and fine-tuning of an LSTM model, equipped with helper functions for efficiency. Finally, it discusses the evaluation of the model using metrics such as a confusion

matrix and classification report. The next chapter will delve into the results and performance of the experiments by the LSTM model in the context of the CHILDES corpora.

Chapter 4

Results

This chapter will provide a comprehensive presentation and analysis of the empirical findings from the research experiments conducted as part of this study. The findings derived from the LSTM model, supplemented with embeddings from both BERT-uncased and RoBERTa models, will be juxtaposed and thoroughly assessed, incorporating the modifications undertaken for this specific experiment. A summative review of the results and key findings will be delivered at the conclusion of the chapter, accompanied by a thoughtful discussion of the implications and insights drawn from the experimental outcomes. The overarching aim is to scrutinize the results in the context of the study's objectives.

4.1 Findings and Experiments

The experiments on the LSTM neural network were conducted by extensive parameter research to achieve the best possible performance. The network provided marginally better results overall with the BERT-uncased embeddings, as well as faster results.

4.1.1 Fine-tuning

The process of fine-tuning was performed through repeated grid-searching. This is a useful method that enables the training process to be repeated for each combination defined by the grid and to narrow down the best combination of hyperparameters to achieve a good model. The selected hyperparameters for the grid search were:

- Learning rates [x, y, z]
- Hidden size [x, y, z]
- Number of layers [x, y, z]

The model used the grid search to save the best state model found and produces the trending values for all combinations, to indicate if further fine-tuning is necessary. The grid search drastically increased the run time. Therefore, when it was found that a parameter repeatedly appeared in the best state model, that value was chosen, and the grid was shortened. The grid used in this thesis was a $3x3x3$ which resulted in 27 iterations of the training process, each time it was running. In Appendix 6.2, you can find a screenshot of the code used in this study.

4.1.2 Hardware

The computer used for training. It uses an Nvidia graphics card that supports CUDA for GPU accelerations to save the training duration.

CPU	AMD Ryzen 7 1800
RAM	32GB
GPU	Nvidia 2070 super

Table 4.1: Caption

4.1.3 Evaluation

A method to evaluate the results can be done by reviewing the confusion matrix, and classification report. The classification report provides information regarding Precision, Recall, F1-Score, Accuracy, Macro avg, and Weighted avg. The confusion matrix displays information on the True Positives, True Negatives, False Positives, and False Negatives. As mentioned in Chapter 2. background, these evaluations can be used to further calculate the model's performance on an unbalanced dataset.

4.1.4 Classification Reports

Below are the classification report provided from the best state model found. As mentioned in chapter 2. background, a way to evaluate the model's performance with an imbalanced dataset, is to calculate the weighted metric done by adding the unweighted F1-Scores, and dividing by the total number of classes. To reiterate, this method treats all classes equally regardless of support.

	Precision	Recall	F1-Score	Support
Class 0	0.41	0.62	0.49	220
Class 1	0.41	0.40	0.41	287
Class 2	0.53	0.34	0.42	238
Class 3	0.49	0.61	0.54	238
Accuracy			0.46	1236
macro avg	0.46	0.50	0.46	1236
Weighted avg	0.47	0.46	0.45	1236

Table 4.2: Classification Report: BERT

	Precision	Recall	F1-Score	Support
Class 0	0.36	0.52	0.43	429
Class 1	0.39	0.45	0.42	550
Class 2	0.63	0.19	0.30	861
Class 3	0.38	0.69	0.49	393
Accuracy			0.41	2233
macro avg	0.44	0.46	0.41	2233
Weighted avg	0.48	0.41	0.39	2233

Table 4.3: Classification Report: RoBERTa

BERT setup:

$$\textbf{Weighted metric} = + \frac{0.49 + 0.41 + 0.42 + 0.54}{4} = 0.465$$

RoBERTa setup:

$$\textbf{Weighted metric} = + \frac{0.43 + 0.42 + 0.30 + 0.49}{4} = 0.41$$

Here it is clear that the BERT setup is outperforming the RoBERTa setup by 0.055

4.1.5 Confusion Matrix

Below are the confusion matrices from the best state model for each setup. These provide information on the predicted values, and the actual values predicted. As mentioned in Chapter 2. a way to evaluate a multiclass imbalanced model is to compute the balanced accuracy. This entails adding the Sensitivity of all classes and dividing by the number of classes.

117	44	33	6
102	116	56	13
88	100	168	135
11	20	61	146

Table 4.4: Confusion Matrix: BERT

224	140	13	52
183	247	34	86
179	204	167	311
30	44	49	270

Table 4.5: Confusion Matrix: RoBERTa

Sensitivity/Recall:

$$\frac{TruePositive}{TruePositive + FalseNegative}$$

BERT setup

$$\textbf{Balanced accuracy} = \frac{0.62 + 0.40 + 0.34 + 0.61}{4} = 0.4925$$

RoBERTa setup

$$\textbf{Balanced accuracy} = \frac{0.52 + 0.45 + 0.19 + 0.69}{4} = 0.4625$$

Again, there is a minuscule increase in performance by the BERT setup, precisely 0.03

4.1.6 Precision-Recall Curve

Is a metric tool to evaluate the output quality from the classifier, when the datasets are very imbalanced. It displays the tradeoff between precision and recall of different thresholds. If the area under the graph is high, then both the precision and recall are high, meaning low false positive and false

negative rates. Additionally, if the precision is low, but the recall is high, then many results will be returned but most will be incorrectly labeled. The opposite will happen if the precision is high, but the recall is low, returning very few results, but most will be labeled correctly [42].

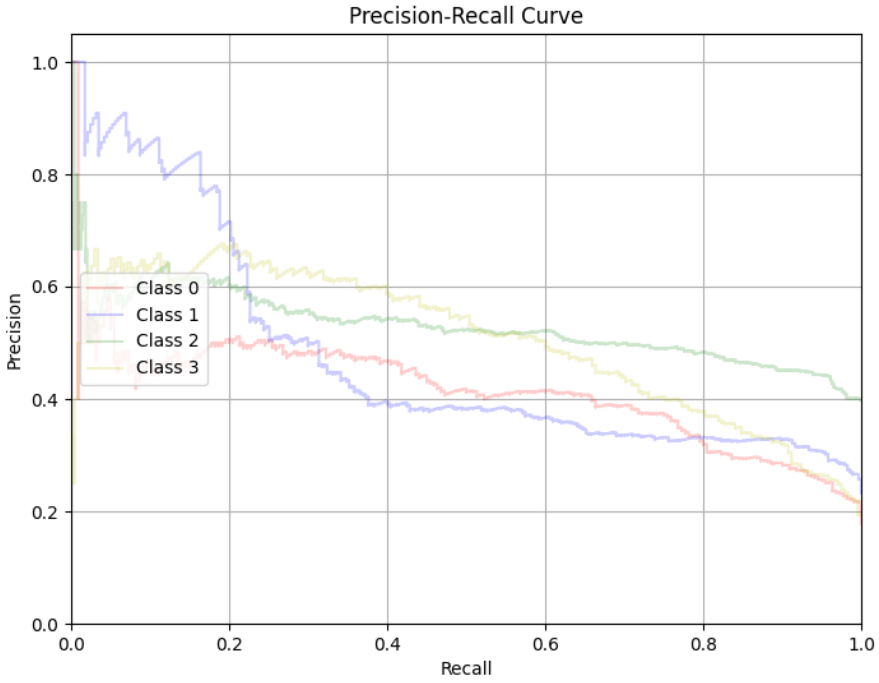

Figure 4.1: Precision-Recall Curve with BERT

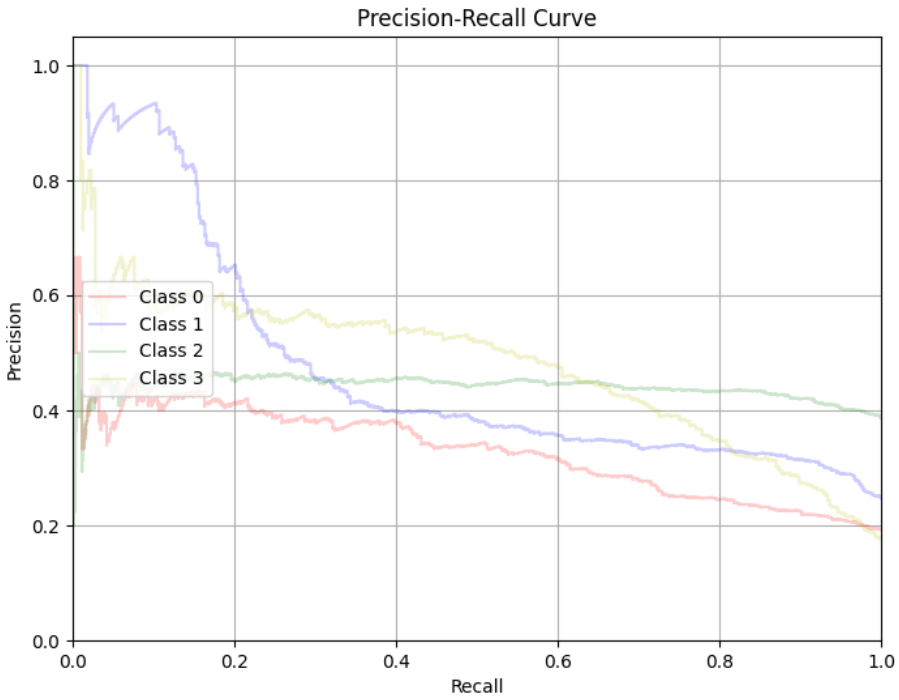

Figure 4.2: Precision-Recall Curve with RoBERTa

By observing both graphs, it is clear to see that all classes provided by the BERT setup have a higher precision starting point, and trend downward with higher recall, compared to the RoBERTa setup. There is a breaking point at around precision = 0.6, and recall = 0.2 where the graph flattens with a downward trend, for both models.

The downward trend after the breaking point in the Precision-Recall curve could indicate that the classifier's ability to correctly identify positive samples (precision) begins to decrease as it attempts to recover more positive instances (increasing recall). Essentially, as the classifier broadens its scope to capture more true positive results (increasing recall), it begins to include more false positives, thereby lowering precision.

The flat segment in the curve suggests that increasing the recall beyond this threshold (0.2 in this case) results in a larger number of false positives, hence the decrease in precision. The classifier is beginning to incorrectly label negative instances as positive, resulting in a larger number of false

positives and consequently, a decline in precision.

This observation points to a trade-off between precision and recall, which is a common challenge in many classification problems. When the model becomes more inclusive to recover more true positives (higher recall), it tends to include more false positives, which lowers precision.

4.2 Effect of data augmentation

After concluding that the best results were found with the BERT setup, we can briefly look into the effects the data augmentations provided. Previous training with the original dataset provided slightly worse results when calculating the weighted metric and balanced accuracy of the non-augmented setup.

182	16	0	22
187	50	0	50
224	32	0	235
54	5	0	179

Table 4.6: Confusion Matrix: BERT without data augmentation

	Precision	Recall	F1-Score	Support
Class 0	0.28	0.83	0.42	220
Class 1	0.49	0.17	0.26	287
Class 2	0.00	0.00	0.00	491
Class 3	0.37	0.75	0.49	238
Accuracy			0.33	1236
macro avg	0.28	0.44	0.29	1236
Weighted avg	0.23	0.33	0.23	1236

Table 4.7: Classification Report: BERT without data augmentation

$$\textbf{Balanced accuracy} = \frac{0.83 + 0.17 + 0.00 + 0.75}{4} = 0.4375$$

$$\textbf{Weighted metric} = + \frac{0.83 + 0.17 + 0.00 + 0.49}{4} = 0.3725$$

As shown, the augmented dataset provided a slight improvement to the overall performance of the best state model.

- The balanced accuracy improved by 0.4925 - 0.4375 = 0.055
- The weighted metric improved by 0.465 - 0.3725 = 0.0925

4.3 Summary

This chapter has offered a comprehensive overview of the experimental findings from this study. The performance of the LSTM model, supplemented with the embeddings from both BERT-uncased and RoBERTa models, has been carefully assessed and juxtaposed, taking into consideration the specific modifications undertaken for the experiment.

Through extensive parameter research and fine-tuning via repeated grid-searching, optimal model performance was pursued. The hardware utilized was capable of providing the necessary computational power, allowing for the efficient execution of these complex processes. The key metrics of model evaluation were meticulously examined through the confusion matrix and the classification report, shedding light on the model's performance, especially within the context of an unbalanced dataset.

Contrasting the classification reports of both the BERT and RoBERTa setups highlighted that BERT outperformed RoBERTa by a slight margin.

This trend was consistent across both the weighted metric and balanced accuracy evaluations. Similarly, the precision-recall curves revealed a higher precision starting point for all classes in the BERT setup, further affirming its superior performance.

Notably, while both setups produced similar hyperparameters after fine-tuning, they varied in the number of layers used, with the BERT setup leveraging three layers, while the RoBERTa setup utilized two. This observation may serve as a critical insight for future explorations in model optimization and refinement.

4.4 Limitations

While the results of this study offer significant insights, it is equally crucial to acknowledge the limitations encountered during the process. Firstly, the performance of our LSTM model, with a validation accuracy of 43.46%, does not measure up to some results reported in prior research related to age detection, as mentioned in the background chapter. Previous studies have reported achievements of 22.22% accuracy using an LSTM model and even up to 65% accuracy employing age recognition through both image and text analysis via LSTM. Although our model's accuracy exceeds the lower range of these comparative studies, it does not reach the higher levels of accuracy reported. This discrepancy underscores the challenge and complexity of age detection tasks and suggests that there may still be room for performance optimization of our LSTM model.

Secondly, the study faced challenges relating to the nature of the dataset used. In particular, datasets containing child conversations, which are crucial for age detection studies, are not commonly available. This scarcity restricts the quality and quantity of data that can be used for such research, potentially limiting the performance and generalizability of the resulting models. Within our dataset, an imbalance was noted, with classes 0 and 1 (representing ages 3 and 4) being overly represented and class 3 being

the least represented. This imbalance might have influenced the model's performance, potentially favoring the prediction of the more represented classes and impeding the effective learning from the underrepresented ones. This calls for caution in generalizing the results and further underscores the need for balanced datasets in training robust and unbiased machine learning models.

Lastly, the duration of the training process also posed a significant limitation. The LSTM model's training on the existing hardware was relatively time-consuming, taking approximately 1.5 to 2 hours per run. This long training duration had implications on the overall efficiency of the fine-tuning process, limiting the number of iterations that could be feasibly executed within a reasonable timeframe. Consequently, this might have impacted the extent to which the model could be optimized.

Recognizing these limitations not only lends a critical perspective to our findings but also highlights potential areas for improvement in subsequent research endeavors. Possible ways to address these limitations could include using balanced datasets, deploying more efficient hardware or employing techniques to speed up the training process, and exploring other methods or techniques that have demonstrated higher accuracy in related research.

Chapter 5

Discussion

5.1 General discussion

One of the significant takeaways from this research is the potential value of advanced linguistic features tailored to the language patterns of children. While the MRC Psycholinguistic Database and CHILDES dataset provided valuable insights, they fall short of capturing the full complexity of children's language development and usage. Future models could benefit from exploring semantic complexity, syntactic structures, and phonological patterns prevalent in child-directed speech, as well as the role of pragmatics and sociolinguistic context.

Furthermore, the importance of larger, more comprehensive, and balanced datasets is undeniable. This study faced challenges with data truncation and outdated word frequencies. More modern and representative datasets could drastically enhance the learning and generalization capabilities of the AI model.

In terms of model selection, this research employed Long Short-Term Memory (LSTM) networks. However, other models like Transformer models or Gated Recurrent Units (GRUs) could present unique benefits. For example, Transformer models excel at handling long-term dependencies, while

GRUs, due to their simpler architecture, may require less computational resources, providing a viable alternative for sequence-related tasks.

Relating these findings back to developmental theories in the background chapter, Piaget's cognitive development theory suggests that children undergo distinct stages of cognitive development, each characterized by unique ways of thinking and understanding. This aligns with the observations made in this study where a one-size-fits-all approach to classifying text may not be effective. Each child may not only be at a different stage but also progress through these stages at varying paces, influencing their language acquisition and comprehension.

This variability could potentially explain the broad age ranges (10-year increments) observed in related research detail in the background chapter. It suggests that age-based text classification, particularly for children, is a complex task that requires a nuanced understanding of the individual's cognitive and linguistic capabilities. In essence, a child's cognitive development may not always align with their chronological age, contributing to misclassification in a model trained on an imbalanced dataset.

This view reinforces the importance of taking a more tailored approach, considering the intricacies of child language development, cognitive stages, and individual variability when training AI models for age-specific text classification. Thus, while the research achieved notable insights, it also revealed an avenue of challenges and opportunities for future work in the field.

5.2 Research Objective

5.2.1 Deep Learning for Age-Based Text Classification

The initial research objective aimed to ascertain if artificial intelligence, specifically deep learning algorithms, could accurately classify text appro-

priate for children within the specified age range of 3 to 15. Based on the study's findings, it can be concluded that this objective was partially met.

The LSTM model employed demonstrated a potential in age-based text classification, achieving a significant degree of accuracy. However, the results also indicated the limitations of the model, given that its accuracy might not be ideal for all applications, particularly those that depend heavily on the precision of age-appropriate educational content.

5.2.2 Challenges and Unique Characteristics in Classifying Children's Age

The second objective was an exploration of the key challenges and unique characteristics in classifying children's age, with a particular focus on linguistic patterns, developmental stages, and imbalance in the dataset.

This study emphasized the complexity inherent in age-based text classification, identifying factors such as the considerable variation in linguistic development across the target age range. This variability is intricately linked with the rapid cognitive, linguistic, and social development observed in children.

Furthermore, the study highlighted issues stemming from the imbalanced nature of the dataset used. The imbalance risked biasing the model towards overrepresented age groups, potentially compromising its performance on underrepresented ones. This aspect underlines the crucial role balanced datasets play in developing robust models for age-based text classification.

5.2.3 Evaluation Metrics and Methodologies

The final objective revolved around identifying suitable evaluation metrics for an imbalanced dataset, understanding additional feature requirements, and determining the role of methodologies such as cross-validation in this process.

The study suggested that the imbalance in the dataset necessitates reconsidering conventional performance metrics. Rather than relying on validation accuracy alone, the research indicated the potential value of weighted accuracy and balanced accuracy metrics, which are designed to handle class imbalances.

Cross-validation emerged as a valuable methodological tool, providing a more robust evaluation framework by utilizing multiple train-test splits instead of a singular division.

In terms of features, this research stressed the potential benefits of integrating advanced, age-specific linguistic features into the model. Such enhancements could allow the model to capture the nuanced differences in language development across the age spectrum.

In conclusion, while strides were made in addressing each of the outlined research objectives, the findings also shed light on the inherent complexities and potential areas of improvement in the task of age-based text classification.

5.3 Lessons learned

Jumping into neural networks felt like diving into a new language - intriguing yet complex. With time and patience, however, a sense of familiarity and understanding began to take hold.

The concept of fine-tuning a LSTM model initially seemed straightforward, until the process proved itself to be quite a labyrinth. The task was not only time-consuming but was also like a complex puzzle, with each adjustment leading to new complexities.

Data management was an unexpectedly challenging task. Visualize a large number of files, each carrying important pieces of information. Keeping them organized and readily accessible proved to be a substantial task.

Chapter 6

Conclusion

6.1 Summary and conclusion

The primary aim of this research was to explore if artificial intelligence, specifically deep learning algorithms, could accurately classify texts suitable for children within the specified age range of 3 to 15 years. To do this, a LSTM-based AI model was created and trained on a carefully curated dataset, supplemented with features from the MRC Psycholinguistic database.

However, the results of the model's performance were below expectations, with a balanced accuracy of 0.4925 and a weighted metric of 0.465, showing the model's inability to effectively distinguish between different age classifications in the imbalanced dataset. This highlights the considerable challenges and complexity of the task at hand.

The study revealed multiple challenges when classifying children's age, particularly related to linguistic patterns, developmental stages, and dataset imbalance, which formed the second research objective. The complexities of children's language development and the intricacies of psycholinguistic features added layers of difficulty to the task.

In terms of the third objective, the research underscored the need for spe-

cific evaluation metrics when working with an imbalanced dataset. It demonstrated the potential of weighted accuracy and balanced accuracy as more informative alternatives to conventional accuracy metrics. Additionally, the use of cross-validation served as a tool to combat overfitting and provide a more robust estimate of the model's performance.

In conclusion, while the research objectives were met in terms of uncovering the complexities of age-specific text classification and identifying appropriate evaluation metrics, the performance of the AI model was less than satisfactory. The study underscores the need for further advancements in data collection and algorithm development to improve the effectiveness of AI in this challenging task. The experience provided valuable insights into the challenges of handling imbalanced data, fine-tuning a complex model, and managing extensive datasets, marking a starting point for future exploration in this field.

6.2 Future work

Looking forward, there are several potential avenues for future research in the field of age detection from children's text. One area of focus could be to address the data scarcity and imbalance issues by seeking to gather and curate a larger and more balanced dataset. This could involve engaging in extensive data collection from diverse sources, potentially also including non-English datasets to expand the scope of the study.

The role of feature engineering in enhancing model performance could also be explored further. For instance, additional or more advanced linguistic features reflecting children's cognitive development stages could be incorporated into the model, aiming to capture the nuances of children's language use more effectively.

In terms of model optimization, future studies could delve into exploring more sophisticated or innovative machine learning and deep learning models that might deliver better performance. Techniques to speed

up the training process, such as utilizing more efficient hardware or employing strategies like transfer learning or federated learning, could also be examined.

Additionally, future research might explore the integration of multimodal data, such as combining text with auditory or visual information, to create more robust and accurate age detection systems. As this field evolves, there is a significant opportunity to push the boundaries of what is currently achievable, paving the way for more precise, reliable, and efficient age detection models.

Bibliography

[1] Emily Alsentzer et al. "Publicly available clinical BERT embeddings." In: *arXiv preprint arXiv:1904.03323* (2019).

[2] Dilanthi Amaratunga et al. "Quantitative and qualitative research in the built environment: application of "mixed" research approach." In: *Work study* 51.1 (2002), pp. 17–31.

[3] Yusuf Arslan et al. "A comparison of pre-trained language models for multi-class text classification in the financial domain." In: *Companion Proceedings of the Web Conference 2021*. 2021, pp. 260–268.

[4] Jason Brownlee. *A Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. 2023. URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (visited on 06/06/2023).

[5] Jason Brownlee. *Understand the Dynamics of Learning Rate on Deep Learning Neural Networks*. 2023. URL: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/ (visited on 06/06/2023).

[6] James P Byrnes. "Piaget's cognitive-developmental theory." In: *Encyclopedia of infant and early childhood development* 87 (2008), pp. 543–552.

[7] CHILDES. *English North American Corpus*. 2023. URL: https://childes.talkbank.org/access/Eng-NA/ (visited on 06/06/2023).

[8] CHILDES. *English UK Corpus*. 2023. URL: https://childes.talkbank.org/access/Eng-UK/.

[9] CHILDES. *TalkBank: CHILDES Database*. 2023. URL: https://childes.talkbank.org/access/.

[10] Shawn L Christiansen and Rob Palkovitz. "Exploring Erikson's psychosocial theory of development: Generativity and its relationship to paternal identity, intimacy, and involvement in childcare." In: *The Journal of Men's Studies* 7.1 (1998), pp. 133–156.

[11] Kevin Clark et al. "What does bert look at? an analysis of bert's attention." In: *arXiv preprint arXiv:1906.04341* (2019).

[12] DeepAI. *Hidden Layer in Machine Learning*. 2023. URL: https://deepai.org/machine-learning-glossary-and-terms/hidden-layer-machine-learning (visited on 06/06/2023).

[13] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv preprint arXiv:1810.04805* (2018).

[14] Exposing.ai. *VGG Face*. [Accessed: May 9, 2023]. 2023.

[15] Stephen Frosh. "A brief introduction to psychoanalytic theory." In: (2012).

[16] Alex Graves and Alex Graves. "Long short-term memory." In: *Supervised sequence labelling with recurrent neural networks* (2012), pp. 37–45.

[17] Syed Zohaib Hassan et al. "Towards an AI-Driven Talking Avatar in Virtual Reality for Investigative Interviews of Children." In: *2nd edition of the Game Systems Workshop (GameSys '22)* (2022).

[18] Iamirmasoud. *Understanding Micro, Macro, and Weighted Averages for Scikit-Learn Metrics in Multi-Class Classification With Example*. 2022. URL: http://iamirmasoud.com/2022/06/19/understanding-micro-macro-and-weighted-averages-for-scikit-learn-metrics-in-multi-class-classification-with-example/ (visited on 05/08/2023).

[19] IBM. *Natural Language Processing*. https://www.ibm.com/cloud/learn/natural-language-processing. 2023.

[20] PyTorch Ignite. *ignite.handlers.early_stopping.EarlyStopping*. 2023. URL: https://pytorch.org/ignite/generated/ignite.handlers.early_stopping.EarlyStopping.html (visited on 06/06/2023).

[21] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. "PyTorch." In: *Programming with TensorFlow: Solution for Edge Computing Applications* (2021), pp. 87–104.

[22] Barbara J Juhasz. "Age-of-acquisition effects in word and picture identification." In: *Psychological bulletin* 131.5 (2005), p. 684.

[23] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. "Grid long short-term memory." In: *arXiv preprint arXiv:1507.01526* (2015).

[24] Milda Macenaite. "From universal towards child-specific protection of the right to privacy online: Dilemmas in the EU General Data Protection Regulation." In: *New Media & Society* 19.5 (2017), pp. 765–779.

[25] Holbrook Mahn. "Vygotsky's methodological contribution to sociocultural theory." In: *Remedial and Special education* 20.6 (1999), pp. 341–350.

[26] Bonan Min et al. "Recent advances in natural language processing via large pre-trained language models: A survey." In: *arXiv preprint arXiv:2111.01243* (2021).

[27] Sparsh Mittal and Shraiysh Vaishay. "A survey of techniques for optimizing deep learning on GPUs." In: *Journal of Systems Architecture* 99 (2019), p. 101635.

[28] Marjorie Montreuil et al. "A review of approaches, strategies and ethical considerations in participatory research with children." In: *International Journal of Qualitative Methods* 20 (2021), p. 1609406920987962.

[29] Catriona M Morrison, Tameron D Chappell, and Andrew W Ellis. "Age of acquisition norms for a large set of object names and their relation to adult estimates and other variables." In: *The Quarterly Journal of Experimental Psychology Section A* 50.3 (1997), pp. 528–559.

[30] Neptune.ai. *Understanding Balanced Accuracy: When Should You Use It and How*. 2023. URL: https://neptune.ai/blog/balanced-accuracy (visited on 05/08/2023).

[31] Elissa L Newport. "Children and adults as language learners: Rules, variation, and maturational change." In: *Topics in cognitive science* 12.1 (2020), pp. 153–169.

[32] Akhilesh Pal. *BERT Explained: State-of-the-Art Language Model for NLP*. Towards Data Science. 2023. URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[33] Mildred L Patten. "Understanding research methods: An overview of the essentials." In: (2017).

[34] Martine B Powell, Belinda Guadagno, and Mairi Benson. "Improving child investigative interviewer performance through computer-based learning activities." In: *Policing and Society* 26 (2016), pp. 365–374.

[35] Programmathically. *Weight Decay in Neural Networks*. 2023. URL: https://programmathically.com/weight-decay-in-neural-networks/ (visited on 06/06/2023).

[36] PyTorch. *torch.optim.lr$_s$cheduler.StepLR*. 2023. URL: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.StepLR.html (visited on 06/06/2023).

[37] Adil Abdul Rehman and Khalid Alharthi. "An introduction to research paradigms." In: *International Journal of Educational Investigations* 3.8 (2016), pp. 51–59.

[38] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. "A primer in BERTology: What we know about how BERT works." In: *Transactions of the Association for Computational Linguistics* 8 (2021), pp. 842–866.

[39] Aleksandr Sergeevich Romanov et al. "Determining the age of the author of the text based on deep neural network models." In: *Information* 11.12 (2020), p. 589.

[40] Alan P Rudell. "Frequency of word usage and perceived word difficulty: Ratings of Kučera and Francis words." In: *Behavior Research Methods, Instruments, & Computers* 25.4 (1993), pp. 455–463.

[41] scikit-learn. *Cross-validation*. 2023. URL: https://scikit-learn.org/stable/modules/cross_validation.html (visited on 05/09/2023).

[42] scikit-learn developers. *Precision-Recall*. 2023. URL: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html (visited on 06/15/2023).

[43] Mike Seymour, Kai Riemer, and Judy Kay. "Actors, avatars and agents: potentials and implications of natural face technology for the creation of realistic visual presence." In: *Journal of the association for Information Systems* 19 (2018), p. 4.

[44] Connor Shorten, Taghi M Khoshgoftaar, and Borko Furht. "Text data augmentation for deep learning." In: *Journal of big Data* 8 (2021), pp. 1–34.

[45] Chanchal Suman et al. "Gender Age and Dialect Recognition using Tweets in a Deep Learning Framework-Notebook for FIRE 2019." In: *FIRE (Working Notes)*. 2019, pp. 160–166.

[46] Kian Long Tan et al. "RoBERTa-LSTM: A hybrid model for sentiment analysis with transformer and recurrent neural network." In: *IEEE Access* 10 (2022), pp. 21517–21525.

[47] Towards Data Science. *Accuracy, Precision, Recall, or F1?* 2023. URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 (visited on 05/09/2023).

[48] Towards Data Science. *Cross-Entropy Loss Function*. 2023. URL: https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e (visited on 05/09/2023).

[49] Towards Data Science. *Precision and Recall*. 2023. URL: https://towardsdatascience.com/precision-and-recall-88a3776c8007 (visited on 05/09/2023).

[50] UWA Psychology. *MRC Psycholinguistic Database (Dict Interface)*. https://websites.psychology.uwa.edu.au/school/mrcdatabase/uwa_mrc.htm. n.d.

[51] Jincheng Xu and Qingfeng Du. "A deep investigation into fasttext." In: *2019 IEEE 21st international conference on high performance computing and communications; IEEE 17th international conference on smart city; IEEE 5th International conference on data science and systems (HPCC/SmartCity/DSS)*. IEEE. 2019, pp. 1714–1719.

Appendix

6.2.1 Main function of embedding creation

```python
def main(sentence_folder, text_file, output_folder):
    #Load the values from the mrc database
    kf_aoa_values = load_kf_aoa_values(text_file)


    for file_name in os.listdir(sentence_folder):
        file_path = os.path.join(sentence_folder, file_name)
        with open(file_path, 'r') as f:
            sentences = [line.strip() for line in f.readlines() if line.strip()]  # Remove empty lines

        # If the file is empty or contains only empty lines, skip it
        if not sentences:
            print(f"Skipping {file_path} as it is empty or contains only empty lines")
            continue

        embeddings = get_embeddings(sentences, kf_aoa_values)
        output_file = os.path.join(output_folder, os.path.splitext(file_name)[0] + "_embedding.npy")
        np.save(output_file, embeddings)
        print(f"Embeddings saved in {output_file}")
```

Figure 6.1: The main function for embedding creation

6.2.2 Example of grid search

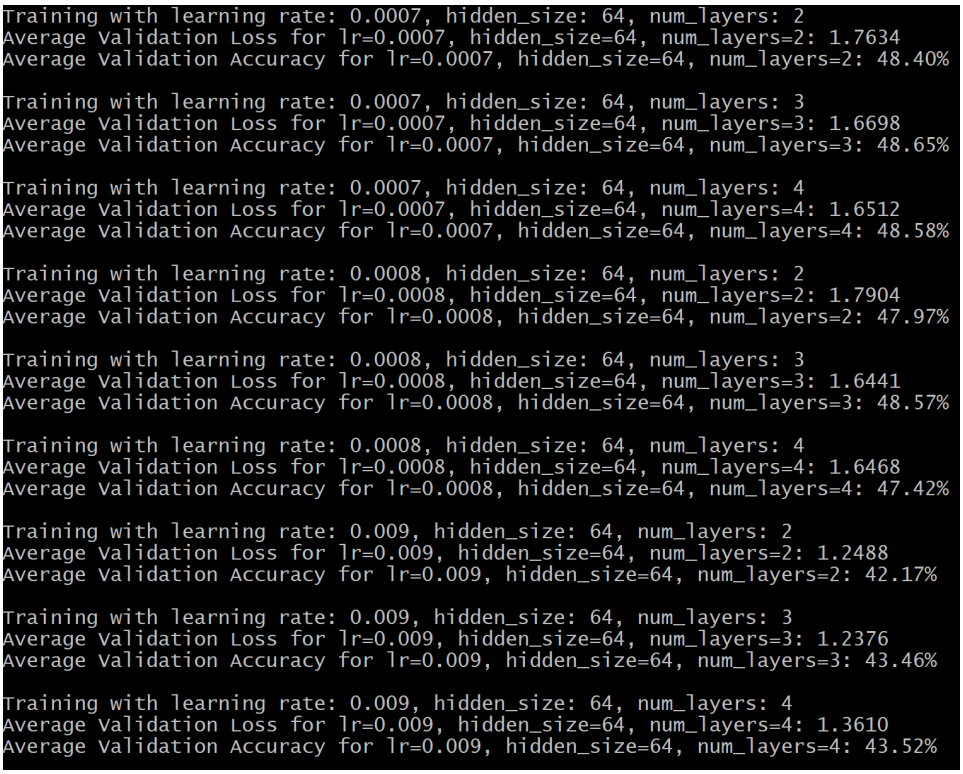

Figure 6.2: Example output from a grid search