UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Machine Learning for Improving Reading Literacy

Larissa H. Tredoux
20783698

Report submitted in partial fulfilment of the requirements of the module Project (E) 448 for the degree Baccalaureus in Engineering in the Department of Electrical and Electronic Engineering at Stellenbosch University.

Supervisor: Dr H. Kamper

October 2020

Acknowledgements

I would like to thank my rabbit, Oreo, for the many hours he spent keeping me company while I worked. I would also like to thank Dr Herman Kamper for letting me truly make this project my own and always being in favour of my suggestions. I could absolutely not have accomplished what I have in this project without his guidance and encouragement.

Figure 1: Oreo, my rabbit, loves to sunbathe nearby me when I work.

UNIVERSITEIT·STELLENBOSCH·UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / *Plagiarism Declaration*

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

20783698 Studentenommer / *Student number*	LH Tredoux Handtekening / *Signature*
LH Tredoux Voorletters en van / *Initials and surname*	06 November 2020 Datum / *Date*

Abstract

English

Learning to read is essential to children's education, but can be a challenging skill to acquire. Most machine learning models are trained on large datasets, but it is difficult to obtain a large set of actual child data. Our aim, therefore, is to construct a machine learning model that analyses small sets of data to predict whether a child would read a sentence correctly or incorrectly, and to generate sentences that the child would likely read incorrectly, to help them improve their reading literacy by practising challenging sentences. To accomplish this, four different types of machine learning models are compared to find the best models for the data: decision trees, logistic regression, k-nearest neighbours, and neural networks. The models are trained on a simulated system of sentences marked "correct" or "incorrect". When comparing the models, decision trees and neural networks produce the best results, displaying an average accuracy of 72.6% and 65.2% respectively. A simple context-free grammar, which is a set of recursive rules that describe string concatenation to form sentences in a language, is implemented to generate new sentences from all sentences classified "incorrect" by these two best models. We conclude that decision trees and neural networks perform the best on smaller datasets, and will therefore result in more relevant sentences being generated by the context-free grammars.

Afrikaans

Om te leer lees is 'n noodsaaklike deel van 'n kind se opvoeding. Dit kan egter 'n uitdaging wees om hierdie vaardigheid aan te leer. Die meeste masjienleermodelle word opgelei met groot datastelle, maar dit is moeilik om 'n groot datastel van werklike kinders te bekom. Ons doel is dus om 'n masjienleermodel saam te stel wat klein datastelle ontleed om te voorspel of 'n kind 'n bepaalde sin korrek sal lees of nie, en om dan nuwe sinne te genereer wat 'n kind waarskynlik verkeerd sal lees, ten einde die kind se leesvaardigheid te verbeter deur uitdagende sinne te oefen. Met die doel voor oë word vier verskillende soorte masjienleer modelle vergelyk om die beste model vir hierdie data te bepaal, naamlik keuse vertakking, logistiese regressie, k-naaste bure en neurale netwerke. Die modelle word afgerig met 'n gesimuleerde stel sinne wat elkeen "korrek" of "verkeerd" gemerk is. Ons vind dat die beste resultate gelewer word deur die keuse vertakking en neurale netwerk modelle, met 'n gemiddelde akkuraatheid van 72.6% en 65.2% onderskeidelik. 'n Eenvoudige konteksvrye grammatika ('n stel rekursiewe reëls wat beskryf hoe stringe tot sinne saamgevoeg word in 'n taal) is geïmplementeer om nuwe sinne te genereer van al die sinne wat as "verkeerd"gemerk word deur hierdie twee beste modelle. Ons gevolgtrekking is dat die keuse vertakking en neurale netwerk modelle die beste presteer met kleiner datastelle, en dus meer relevante sinne sal genereer deur middel van konteksvrye grammatika.

Contents

Declaration ii

Abstract iii

List of Figures vii

List of Tables viii

Nomenclature ix

1. Introduction 1

1.1. Problem statement 1

1.2. Objectives 2

1.3. Methodology 2

1.4. Report layout 3

2. Background 4

2.1. Related studies 4

2.1.1. Human assessment of oral reading errors 4

2.1.2. Automated assessment of oral reading errors 5

2.2. Identifying common reading errors 6

2.3. Discussion 7

3. System design for sentence classification and generation 8

3.1. Agents 8

3.2. Feature vectors 9

3.3. Models 10

3.3.1. Decision trees 10

3.3.2. Logistic regression 11

3.3.3. k-nearest neighbours 12

3.3.4. Neural networks 13

3.4. Sentence generation 14

3.5. Evaluation 15

3.6. Discussion 16

4. System implementation **18**
4.1. Agents . . . 18
4.2. Feature vectors . . . 21
4.3. Models . . . 22
4.3.1. Decision trees . . . 22
4.3.2. Logistic regression . . . 26
4.3.3. k-nearest neighbours . . . 27
4.3.4. Neural networks . . . 27
4.4. Sentence generation . . . 29
4.5. Discussion . . . 31

5. System evaluation **33**
5.1. Analysis of model performance . . . 33
5.1.1. Selecting the best model . . . 33
5.2. Context-free grammar . . . 34
5.3. Discussion . . . 36

6. Summary and Conclusion **38**
6.1. Machine learning for improving reading literacy . . . 38
6.2. Future work and recommendations . . . 40

Bibliography **41**

A. Project Planning Schedule **43**

B. Outcomes Compliance **44**

List of Figures

1. My rabbit, Oreo i

3.1. System development process 8
3.2. Decision tree structure 11

4.1. A comparison of the accuracy, precision and recall of the three decision tree validation models 23
4.2. Decision Tree Two for Agent 7 24
4.3. A comparison of the accuracy, precision and recall of the three logistic regression validation models 27
4.4. A comparison of the accuracy, precision and recall of the three KNN validation models 28
4.5. A comparison of the accuracy, precision and recall of the three neural network validation models 29
4.6. Example context-free grammar 30

5.1. A comparison of the accuracy, precision and recall of the four machine learning models 34
5.2. A comparison of the accuracy, precision, recall, predicted proportion of sentences containing errors and actual proportion of sentences containing errors for the neural network and decision tree models 35

List of Tables

4.1. Errors for each agent, corresponding to specific words or graphemes 20
4.2. The average accuracy, average precision and average recall for each of the decision tree validation models 24
4.3. True labels and predicted labels for a decision tree 25
4.4. The average accuracy, average precision and average recall for each of the logistic regression validation models 26
4.5. The average accuracy, average precision and average recall for each of the k-nearest neighbours validation models 28
4.6. The average accuracy, average precision and average recall for each of the neural network validation models 29

5.1. The average accuracy, average precision and average recall for Decision Tree Three, Logistic Regression One, k-Nearest Neighours Two and Neural Network Two 34
5.2. The average accuracy, average precision, average recall, predicted average proportion of sentences containing errors and actual average proportion of sentences containing errors Neural Network Two and Decision Tree Three . 36

A.1. The goals of the project within a planned time frame 43

B.1. Each ECSA exit level outcomes along with a description of how it has been achieved and which chapters are relevant to it 44
B.2. Each ECSA exit level outcomes along with a description of how it has been achieved and which chapters are relevant to it (cont'd) 45
B.3. Each ECSA exit level outcomes along with a description of how it has been achieved and which chapters are relevant to it (cont'd) 46

Nomenclature

Variables and functions

m	A region corresponding to a leaf node in a decision tree.
k	A class index in $k = 1...K$ for a decision tree classification problem.
$\hat{p}_{\mathrm{m,k}}$	For a decision tree, the proportion of points in region m that belong to class k.
G_m	The Gini index indicating the purity of region m, in a decision tree.
$\mathbf{w}$	The weight vector for a logistic regression model.
$\mathbf{x}$	A variable representing a feature vector for the training set in logistic regression.
w_0	The offset for the decision boundary in logistic regression.
$E(\mathbf{w})$	The negative log-likelihood with respect to $\mathbf{w}$ in logistic regression.
n	A class index in $n = 1...N$ for a logistic regression classification problem.
y_n	A variable representing class labels for logistic regression. $y_n = 1$ and $y_n = 0$ indicate Classes 1 and 2 respectively.
$\boldsymbol{x}_n$	Data sampled from the training set that belongs to class n in logistic regression.
$\sigma(x)$	The sigmoid function with respect to variable x.
λ	The regularisation hyperparameter in logistic regression.
p	The order of the distance in Minkowski distance calculations.
i	The index of an element in an n-dimensional vector, where $i = 1...n$.
x_i	The ith element of an n-dimensional vector $\mathbf{x}$.
y_i	The ith element of an n-dimensional vector $\mathbf{y}$.

TN	A true negative, that is, a sentence that was classified as “correct” and which is, in fact, correct.
TP	A true positive, that is, a sentence that was classified as “incorrect” and which is, in fact, incorrect.
FN	A false negative, that is, a sentence that was classified as “correct” and which is, in fact, incorrect.
FP	A false positive, that is, a sentence that was classified as “incorrect” and which is, in fact, correct.
C	A parameter representing the inverse of regularisation strength for logistic regression.

Acronyms and abbreviations

KNN	k-nearest neighbours
ANN	artificial neural network
MLP	multi-layer perceptrons
NN	neural network
CFG	context-free grammar
S	start-symbol
NP	noun phrase
VP	verb phrase
Det	determinant
Nom	nominal
N	noun
V	verb
PP	prepositional phrase
P	preposition

Chapter 1

Introduction

In South Africa, many students struggle with reading. In fact, in a study that compared 50 countries, South Africa placed last in reading literacy [1]. Learning to read is an essential part of children's education; the acquisition of knowledge is heavily dependent on one's ability to read quickly, accurately and with understanding.

Machine learning can provide useful techniques to help children learn more effectively. For example, machine learning can allow an automated reading tutor system to listen to a child read, use speech recognition to detect reading miscues, and give the child personalised feedback to help them improve their reading [2]. However, most machine learning models are trained on large datasets, and it would be difficult and time-consuming to obtain a large set of actual reading data from children.

Another problem to consider is that many South African languages are low-resource languages, which means that large amounts of speech and language data are simply not available for those languages. This makes it even more challenging to implement traditional machine learning solutions to help children improve their reading in those languages.

1.1. Problem statement

A possible alternative to using the aforementioned traditional machine learning models, which are trained on copious amounts of data, is to develop a machine learning model that can use much smaller amounts of data. Therefore, the aim of this project is to construct an application that uses a machine learning model to analyse small sets of data in order to identify the types of sentences that a first-time reader consistently struggles with. By identifying the properties of the sentences with which the reader struggles most, an algorithm should be able to predict whether a reader will read a sentence correctly or incorrectly.

The application should be tailored to help the reader with their specific issues, by generating sentences that the reader would likely read incorrectly. This will help them improve their reading literacy by practising challenging sentences.

1.2. Objectives

The goals of this project are to (1) construct a variety of different agents, simulating struggling first-time readers, to use in training and testing the machine learning models, (2) compare different machine learning models to find out which performs best when trained and tested on small datasets, and (3) generate sentences that are relevant to the specific issues that the reader experiences while reading, in order for them to practise and improve their reading literacy.

1.3. Methodology

In total, 19 different agents are constructed, each consisting of 100 sentences marked "correct" or "incorrect". The agents are intended to each represent a child that struggles to read certain words or letter combinations. These agents are used in lieu of real children, to eliminate the complicated and time-consuming process of collecting child data. Each agent is assigned a set of between one and four "reading errors", represented by words or letter combinations that the agent would theoretically struggle to read, were they a real child. Sentences that contain the words or letter combinations that the particular agent struggles with are then marked as incorrectly read.

Four different machine learning models are trained on feature vectors constructed from the sentences of each agent. The models used are decision trees, logistic regression, k-nearest neighbours and neural networks. This is where a particular benefit of using agents manifests itself: we know what types of errors these models should be detecting. With real children, this would not be possible.

The decision trees and neural networks perform the best on our small datasets. The decision trees display an average accuracy of 72.6% and the neural networks display an average accuracy of 65.2%, compared to logistic regression, which display an average accuracy of 65.9% and k-nearest neighbours, which display an average accuracy of only 58.5%. Although the logistic regression models display approximately 1% higher accuracy and precision on average than the neural networks, the neural networks display an average recall which was 8% higher than that of the logistic regression models, so the neural networks are considered to be our second-best model.

A context-free grammar is a notation, described by a system of productions – that is, recursive rules – that govern string concatenation in a language. Since strings can be sentences, a context-free grammar can be used to construct sentences in a language, according to the productions. A basic context-free grammar is used to generate sentences, based on the results of the neural networks and decision trees for each agent. The grammars are constructed from the sentences classified as incorrect by those two models, in order to generate sentences that are relevant to the specific reading errors made by each agent.

1.4. Report layout

This report covers background information in the form of discussion of related works, as well as sources used, in Chapter 2. Chapter 3 discusses the design of the system, including agents, feature vectors, the different models, and sentence generation, as well as metrics used in evaluation of the system. In Chapter 4, the implementation of the agents, feature vectors, machine learning models and sentence generation is described in detail. The evaluation of the entire system is described in Chapter 5, and results are discussed. Finally, Chapter 6 gives a summary and conclusions, as well as recommendations.

Chapter 2

Background

Studies regarding the analysis of errors made by early readers can be divided into two broad categories: those that use machine learning, and those that do not. Among those that do not, there are studies that have primarily focused on classifying the errors and identifying trends as in [3] or attempting to understand why the readers were making certain errors as in [4]. Those that do use machine learning, on the other hand, such as [2, 5, 6], focus on the detection, classification, and analysis of the errors but not the remediation of the errors. Furthermore, all the abovementioned studies tend to see children as an aggregate, rather than individuals, and do not consider the fact that every child has different needs when it comes to their education.

2.1. Related studies

2.1.1. Human assessment of oral reading errors

In [3], the author observed and analysed the oral reading errors made by 50 children as they progressed through the latter eight months of the first grade. Her primary objective was to identify various trends among the children, namely what types of errors they were making, how they used graphic information, and how they used grammatical information during oral reading. Through these analyses, she aimed to gain an improved understanding of reading as a cognitive response to sequential visual symbols.

To accomplish the above, she collected reading data from 50 first-grade students from the same school. The students were taught letter sounds and spelling patterns. Every month for the eight-month duration of the study, the children were examined on their reading and their errors were noted and classified according to a system of seven different types of errors. Good and poor readers were identified from the results of the monthly reading examinations.

The study found that the types of errors made by the children evolved over time, with the good readers and poor readers exhibiting different types of errors. Good readers displayed an improved graphic and grammatical accuracy, while poor readers relied on the first or last letters of words in order to identify them.

This study in particular focused primarily on trends among the readers, and attempted to classify the children into groups of “good” and “poor” readers. This is a very generalised approach, and while it can be helpful to identify trends among groups of children, there will undoubtedly be many outliers to whom these trends do not apply at all. Basing recommendations on general observations means that those children who face challenges when reading that are different from the general case will inevitably be at a disadvantage in their education.

The objectives of [4] were very similar. Again, a sample of 21 first-grade children from a single school was examined to determine which reading errors were common among the children. This study, however, aimed to identify the features of the errors that were similar to the correct responses, in an effort to gain insight into the reading strategies of early readers.

The 21 children were observed throughout their school year and their oral reading errors were recorded in writing. Errors were classified into one of four different categories – substitution, omission, insertion, and reversal or scrambling – and children were divided into groups of high achievers and low achievers based on how quickly they progressed in their reading abilities. The similarity of the error to the correct response was calculated according to metrics such as the graphic similarity of the words, as well as shared letters.

The author of this study discovered that all the children displayed an improved reading level by the end of the school year. The children employed their knowledge of linguistic structure in order to identify words. As expected, the features that were common between the errors and the correct responses, such as shared letters, were indicative of the reading strategies used by the children.

Once again, this study focused on grouping children based on their reading ability, and identifying what the children had in common, rather than focusing on any differences in their reading strategies that may have been impacting their reading abilities.

Both of the studies described above aimed to identify common reading errors among the children, and aimed to classify the children themselves as “good” or “bad” readers, without actively trying to find a solution to help the poorer readers improve. In addition, they sought rather to classify errors, rather than observe specific words or phrases that the children struggled with. The approach used by these studies is not particularly helpful when it comes to our machine learning application, since the algorithm that will be implemented in this project will be analysing the common features of sentences that the child is struggling with, rather than the types of errors the child is making.

2.1.2. Automated assessment of oral reading errors

In contrast to [3] and [4] which make use of exclusively human observers to record children’s reading errors, [5] aims to detect reading miscues and disfluencies automatically, comparing the performance of the algorithms against human evaluators to assess their similarity.

The objective of the authors was to automatically assess a group of 255 predominantly Mexican-American children, in Kindergarten and First and Second Grade, to determine their level of English literacy, by detecting reading miscues and disfluencies. To achieve this, the children each read 55 words of varying complexity, and the types of reading miscues they exhibited were noted. Human evaluators were used as a benchmark to determine which disfluencies were the most significant in terms of accuracy and fluency. An automatic disfluency detector was developed based on the results obtained from the human evaluators and used to create a classification algorithm to determine whether a particular utterance could be considered fluent or disfluent.

The authors found that the disfluency detector could successfully detect different types of disfluencies, with a 14.9% missed detection and 8.9% false alarm rate.

This study shows that it is indeed possible for children's reading errors to be automatically detected. However, the corpus used by the authors in developing their algorithms consisted of nearly 30 000 recordings and took much time and effort to compile [7]. In addition, the aim of the project was merely to detect reading children's reading disfluencies, and not to attempt to help children improve their reading skills.

Another study that includes automated miscue detection during oral reading is [2]. In this paper, the authors attempt to gain an increased understanding of why speech recognition errors occur, in addition to developing a system to detect miscues. This paper contributed to the Colorado Literacy Tutor program, aimed at helping children improve their reading.

For their study, the authors made use of a corpus consisting of around 10 000 words, and developed a new labelling system to facilitate improved word error analysis. They also created a framework to detect oral reading miscues, implemented with the help of a linear classifier as well as features from a speech recogniser.

The authors achieved a 67% success rate when detecting reading miscues with their automated miscue detector. Their labelling scheme was, to their knowledge, unprecedented. However, the corpus that they used was fairly large and their study did not directly contribute to mitigating the struggles children face when learning to read.

2.2. Identifying common reading errors

All the above studies focused on identifying, analysing and classifying oral reading miscues. In contrast, [6] focuses on common errors that children make in the process of mapping graphemes to phonemes. Graphemes are letters or groups of letters that represent sounds, and phonemes are the smallest sound units in a language.

The authors' goal was to provide insight into which graphophonemic contexts are especially likely or unlikely to engender oral reading miscues. This was intended to provide useful information to automated reading tutors, so that they can better predict children's reading miscues.

From a database comprising over 70 000 oral reading mistakes, the authors mapped the graphemes in the mistakes to the correct and spoken corresponding phonemes. A discrepancy between the correct and spoken corresponding phonemes was considered a decoding error. They then employed a decision tree algorithm to determine 225 positive rules that predicted the frequency of the 71 most common decoding errors in different graphophonemic contexts.

To determine how general the 225 rules were, the authors tested how effectively they predicted the frequency of identical decoding errors when the readers and text were different. The correlation between predicted and actual frequencies was found to be 0.473, while the context-independent versions of the rules achieved a correlation of 0.350.

2.3. Discussion

Three conclusions can be drawn from the literature discussed in this chapter.

The first is that reading studies often focus on only the problems faced by children when learning to read. Researchers aim to classify mistakes and to understand why children are making certain mistakes. Illuminating the problem can be a first step in finding a solution, but does not in itself help children improve their reading.

The second is that children are viewed, not as individuals who face individual challenges, but as mere data sources, that either perform according to a certain standard, or do not. This is evident when researchers classify children as either “good” or “poor” readers. While this can contribute to understanding difficulties that children in general experience when learning to read, a lack of personalisation implies that any individuals who have reading difficulties that diverge from the general case will be marginalised.

The third is that large corpora are generally, if not always, used when analysing children’s oral reading miscues. Of course, the larger the dataset, the more reliable the results will be in most cases. However, it is simply not feasible in some instances to collect such volumes of child data, either because it requires large quantities of effort and time, or because the language itself does not permit it, as is the case for low-resource South African languages.

As the reader will recall from Chapter 1.1, this project specifically focuses on helping children improve their reading literacy with an application that is tailored to help the reader with their unique challenges, and aims to do so with the help of a small set of data. These are goals that have not been achieved in the related studies, even though contributions were made to systems that do help improve children’s literacy, in [2, 5, 6].

One study that is helpful for this project specifically is [6]. Chapters 3 and 4 will describe how [6] is used in constructing agents with which to assess the different machine learning models used in this project.

Chapter 3

System design for sentence classification and generation

The design of the system which classifies sentences as "correct" or "incorrect" and then generates new sentences, is a four-step process. First, the agents have to be designed to simulate children who read a sentence either correctly or incorrectly. Second, the agents have to be translated into feature vectors that could be interpreted by machine learning models. Third, there is the question of which machine learning models to use. The final step is to devise a method for generating new sentences based off the predictions of the machine learning models. Figure 3.1 illustrates the sequence in which the system is developed.

In addition to designing the system, it is necessary to decide on metrics that will be used to measure the performance of the different machine learning models.

3.1. Agents

Since the system developed in this project is intended to help first-time readers improve their reading skills, the first step in the design process is to either collect data from readers or simulate reading data. By nature, this project requires fairly small quantities of data, but, even so, obtaining enough validation and test data from real children would have required a significant amount of time and work, not to mention the difficulty of finding between ten and 20 children to participate in a study in the middle of a nation-wide lockdown. Therefore, we make the decision to use a variety of text-based agents, representing artificial children that read sentences incorrectly or correctly, on which to train the machine learning models.

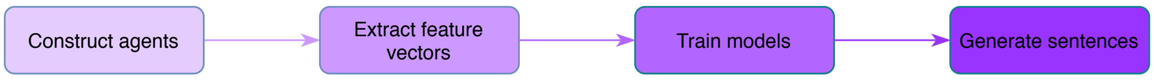


Figure 3.1: The process of developing the system used to classify sentences and generate additional sentences includes constructing agents, extracting feature vectors from those agents, training models on those feature vectors, and generating new sentences from all the sentences that are classified incorrect by the model.

One of the aims of this project is to identify specifically the types of sentences that individual first-time readers have trouble reading. Bearing this in mind, the agents are designed as a collection of sentences, labelled according to whether they have been “read” correctly or incorrectly. Each agent will, with a high level of consistency, make one or more “reading errors”, which will hopefully be recognised by the machine learning models in a later part of the project. The reading errors are based on graphemes that could be read incorrectly by a child, as inspired by [6]. If the pertinent graphemes are present in a particular sentence, then the child has a high probability of reading that sentence incorrectly, and, consequently, that sentence can be labelled as having been read incorrectly. Each child is different, and no two children make exactly the same reading errors, and therefore each agent is designed to make unique errors.

A particular advantage of testing the models on text-based agents rather than actual children is that the cause of the agents’ errors is known; in addition to knowing which sentences the agent is reading incorrectly, we know why they are reading those sentences incorrectly. This is especially useful for testing, as it allows us to ensure that the machine learning models are indeed detecting the errors the agent is making. In contrast, with an actual child, we would not know the underlying systematic mistakes that they make, and would be limited to knowing only which sentences they read incorrectly.

3.2. Feature vectors

While a human can read and understand text sentences, a machine learning model cannot. So, the next step in the design process is to devise a method whereby each sentence of each agent can be converted into a feature vector that can be understood by a machine learning model. The requirement for this is that the features in the feature vector should allow a machine learning model to easily recognise sentences that are similar to those labelled as incorrectly read.

For the purpose of obtaining a feature vector from each sentence, we consider a sentence to be simply a collection of words, and decide to use a bag-of-words method. This is a straightforward method that considers all the words that are present in all the agents, and examines each sentence to see which of the words are present in that particular sentence. Since sentences that contain the same words can be considered similar, this also allows the machine learning models to easily recognise which words are similar, by merely noting which words they have in common, which is useful for classification. The bag-of-words method also disregards the ordinal properties of the words in each sentence. As a result, sentences that contain the same words, but in a different order, would still be recognised as similar. This is both an advantage and a disadvantage – a child may struggle with certain words regardless of the order they are in, but may also mispronounce a word only when it appears in a certain order with certain other words.

A further rationale for a bag-of-words method is that the "reading errors" that are simulated in the agents depend on graphemes, and if two sentences contain the same word, then they naturally also contain the same graphemes, which means that the reader is likely to make the same error in both of those sentences. The bag-of-words method for extracting feature vectors facilitates easy recognition of such cases. An example of feature extraction using a bag-of-words method is given in Chapter 4.2.

3.3. Models

In the interest of determining what type of machine learning model is best suited to small datasets, four different models are compared in this project. To achieve a thorough comparison, the models are chosen to be as diverse as possible. The models that we implement are decision trees, logistic regression, k-nearest neighbours and neural networks.

The purpose of the models is to determine which sentences are similar to those which have been labelled "incorrect" for each respective agent, and classify these sentences as "incorrect" while classifying the other, non-similar or less similar, sentences as "correct". This approach assumes that the agent makes only a limited amount of errors, consistently. This may or may not be an accurate representation of a real child struggling to read, but it is judged to be an appropriate approximation.

3.3.1. Decision trees

Decision trees are intuitive, since they are similar to how a human makes decisions, checking if a feature meets certain criteria in order to classify it. A human evaluator would listen to a child reading, conclude that they struggle with certain types of sentences with common words or graphemes, and then identify similar sentences that the child would likely read incorrectly. Similarly, a decision tree identifies particular features common to sentences that the reader struggles to read. Then, if those features are present in other sentences, the algorithm classifies those sentences as likely to be incorrectly read also.

Decision trees are used for classification problems. An example structure of a decision tree is shown in Figure 3.2, showing the root node at the top, an internal node, and three leaf nodes, or leaves, which are nodes that result in no further splits in the data. Given training vectors and labels, a decision tree recursively splits its nodes with the aim of grouping samples with the same labels together in the leaves [8]. This process is called growing the tree.

The tree-growing algorithm begins at the root node, in a top-down approach. To split each node into further nodes, it steps through each possible split point for each feature, and calculates the reduction in leaf impurity that would result from splitting at that point in the data. A greedy algorithm selects the best splitting point, according to which split

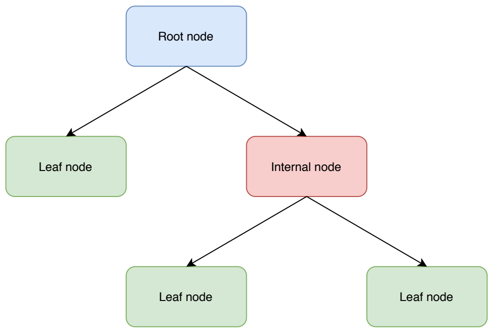


Figure 3.2: A typical decision tree consists of one root node at the top, as well as internal nodes and leaf nodes.

gives the greatest reduction in leaf impurity, and two new nodes are created. This process repeats until pre-specified stop conditions are met.

Leaf purity, which corresponds to the proportion of points in the region represented by a leaf, is measured by the Gini index, which can be calculated for a region m using Equation 3.1 [9].

Here, $\hat{p}_{\mathrm{m,k}}$ represents the proportion of points in region m that belong to class k, where, assuming we have K classes, $k = 1...K$.

$$G_m = \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k}) \tag{3.1}$$

For a two-class system, as used in this project, we have $k = 1, 2$. This results in the proportions $\hat{p}_{\mathrm{m,1}}$, and $\hat{p}_{\mathrm{m,2}}$, leading to Equation 3.2. Therefore, we can use a simplified equation for the Gini index for a two-class system, as illustrated in Equation 3.3.

$$(1 - \hat{p}_{m,1}) = \hat{p}_{m,2} \tag{3.2}$$

$$G_m = \sum_{k=1}^{2} \hat{p}_{m,1}\hat{p}_{m,2} \tag{3.3}$$

3.3.2. Logistic regression

Binary logistic regression is an algorithm which can be implemented for dichotomous classification problems. As our classification problem is, indeed, a dichotomous classification problem – having two possible outcomes: “correct” and “incorrect” – we choose to also consider this approach for sentence classification.

Where decision trees simply analyse which features are present for a particular item of data and which are not, logistic regression aims to maximise the likelihood that the data item in question belongs to a certain class, or, in other words, determine whether it is more likely that a sentence will be read incorrectly as opposed to correctly.

Contrary to what its name suggests, logistic regression is a classification model and not a regression model. Binary logistic regression aims to calculate a decision boundary between two classes in such a way that maximal separation is achieved between these classes. Maximising the separation between the two classes means that all data items are as far from the decision boundary as possible, which implies that there is as little doubt as possible about which class a data item belongs to. The decision boundary, described in Equation 3.4, is determined by the weight vector, $\mathbf{w}$.

$$\mathbf{w}^T\mathbf{x} + w_0 = 0 \tag{3.4}$$

To achieve maximal class separation, and thereby maximise the probability that a data item belongs to a particular class, the logistic regression algorithm minimises the negative log-likelihood, $E(\mathbf{w})$, which is defined in Equation 3.5 [10]. For a two-class system, $y_n = 1$ and $y_n = 0$ indicate Classes 1 and 2 respectively, $\mathbf{w}$ represents the weight vector, $\sigma()$ is the sigmoid function, and $\boldsymbol{x}_n$ is a data value.

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\{y_n \ln\sigma(\mathbf{w}^T\boldsymbol{x}_n) + (1-y_n)\ln(1-\sigma(\mathbf{w}^T\boldsymbol{x}_n))\} \tag{3.5}$$

The optimal parameters for the negative log-likelihood function can be calculated using gradient descent methods, such as the Newton-Raphson method or quasi-Newton methods [11]. Equation 3.6 gives the gradient of the negative log-likelihood function, and Equation 3.7 gives the system of equations for calculating the weight vector $\mathbf{w}$.

$$\Delta E(\mathbf{w}) = \sum_{n=1}^{N}(\sigma(\mathbf{w}^T\boldsymbol{x}_n) - y_n)\boldsymbol{x}_n \tag{3.6}$$

$$\sum_{n=1}^{N}(\sigma(\mathbf{w}^T\boldsymbol{x}_n) - y_n)\boldsymbol{x}_n = \mathbf{0} \tag{3.7}$$

The shape of the logistic sigmoid function, $\sigma()$, shows that, to achieve maximal class separation, $\mathbf{w}^T\mathbf{x}$ must be large. To prevent merely choosing large values of $\mathbf{w}$, it is necessary to restrict the value of $\mathbf{w}$ through constrained optimisation or by adding a penalty term.

To avoid overfitting or specialisation, it is important to have a regularisation term, expressed as $\frac{1}{2\lambda}\mathbf{w}^T\mathbf{w}$. This regularisation term includes the hyperparameter λ, which is determined during validation after the initial training stage of the model. Instead of λ, we can also use C, a regularisation parameter that represents the inverse of regularisation strength and is inversely proportional to λ.

3.3.3. k-nearest neighbours

k-nearest neighbours (KNN) in its essence calculates whether two items of data are similar, in order to predict which class they belong to. Since that is precisely what we are aiming to do with the sentences in our dataset, it seems logical that the KNN algorithm would be an apt choice.

Unlike decision trees, logistic regression, and neural networks, k-nearest neighbours classification does not have a training phase where it constructs a generalised model from the data [12]. Instead, it analyses the training data directly during classification. This means that k-nearest neighbours is an example of instance-based learning.

In order to determine which class a data item belongs to, the classifier simply inspects all the other data items and determines the k nearest data items, where k is specified by the user. Then, it ascertains which class the majority of those k data items belongs to, and assigns that class to the data item in question. For example, for a data item in a two-class system, where $k = 3$, if two of the three nearest neighbours belong to Class 1 and one belongs to Class 2, the data item will be placed in Class 1.

The distance between data items can be calculated through a variety of different methods, including Minkowski, Manhattan and Euclidean. These three methods will be compared in this project, to ascertain which delivers the best results. Equations 3.8, 3.9 and 3.10 give the formulae for Minkowski distance, Manhattan distance and Euclidean distance respectively. In these equations, p indicates the order of the distance, which is being calculated between n-dimensional data items $\mathbf{x}$ and $\mathbf{y}$. The Manhattan distance is the Minkowski distance where $p = 1$, and the Euclidean distance is the Minkowski distance where $p = 2$.

$$(\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}} \tag{3.8}$$

$$\sum_{i=1}^{n} |x_i - y_i| \tag{3.9}$$

$$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3.10}$$

3.3.4. Neural networks

Neural networks are designed to *very* roughly imitate the functioning of the human brain and are intended to be able to recognise patterns (although this is a contentious issue). This appears to be ideal for a system that must be able to recognise patterns in the sentences that a child is reading correctly or incorrectly.

In this project, we use a type of artificial neural networks (ANN) called multi-layer perceptrons (MLP). MLP can be used for either classification or regression, but this project utilises the MLP classification algorithm. The MLP algorithm is similar to logistic regression, but contains one or more non-linear hidden layers between the input layer and output layer [13]. In each hidden layer, the values from the previous layer undergo a transformation by means of a weighted linear summation and, subsequently, a non-linear activated function. The final hidden layer passes the transformed values to the output layer, which then converts them to output values. To find the optimal values for the weights in the hidden layers, the algorithm aims to minimise the Cross-Entropy loss function (in fact, for the binary case, this is what is given in Equation 3.5), using gradient descent. Backpropagation (propagating the error backwards) is used to calculate the gradients.

For binary classification, as used in this project, the logistic function shown in Equation 3.11 is used to calculate the output values. The function yields values between zero and one, and a positive classification of 1, corresponding to "incorrect", would be assigned to values greater than or equal to one half (0.5), while values less than one half would be assigned a classificaton of 0, corresponding to "correct".

$$g(z) = \frac{1}{1 + e^{-z}} \tag{3.11}$$

To prevent overfitting, a regularisation term, α, is used to penalise weights in the hidden layers that are large in magnitude.

3.4. Sentence generation

Two strategies are considered for the generation of sentences. The first is a Markov chain, and the second is a context-free grammar.

One must consider here that whenever sentences are generated randomly by an algorithm, there is no guarantee that these generated sentences will make sense, unless a highly sophisticated algorithm is used that includes checks for grammar, syntax and semantics. However, constructing such an algorithm is not within the scope of this project.

A Markov chain relies on probabilities to generate sentences. It analyses the given data to calculate the probabilities that a certain next word will occur after the previous word, and generates sentences according to these probabilities [14]. While the results may sometimes be sensible and at least partially grammatically correct, this approach does not take any grammatical, syntactic or semantic correctness into consideration when generating the sentences.

A context-free grammar, on the other hand, corresponds to the production rules for concatenating strings in a formal language [15]. This ensures that the construction of the generated sentences will at least have some semblance of grammatical correctness – they will be correct according to the context-free grammar. They may not make sense, and they may include syntax and concord errors, but they will quite possibly be more correct than any sentences generated by a Markov chain. Therefore, a simple context-free grammar is implemented to generate sentences for this project.

3.5. Evaluation

In order to compare the four machine learning models, three different metrics are used to evaluate their performance. These are accuracy, precision and recall.

To avoid confusion, an important detail to note is that a “positive” classification entails classifying a sentence as “incorrect”, while a “negative” classification entails classifying a sentence as “correct”. This may initially seem counterintuitive, but bearing in mind that the aim is to find out which sentences would be read incorrectly, it is useful to employ this approach.

In Equations 3.12, 3.13 and 3.14 below, the variables TN, TP, FN and FP are used to describe the classification of sentences. TN indicates a “true negative”, that is, a sentence that is classified as “correct” and which is, in fact, correct. Similarly, TP indicates a “true positive”, that is, a sentence that is classified as “incorrect” and which is, in fact, incorrect. FN indicates a “false negative”. This is a sentence that is classified as “correct” but is actually incorrect. Finally, FP indicates a “false positive”, which indicates that a sentence was classified as “incorrect” when it is really correct.

Accuracy can be described as how many sentences are classified correctly by the model, relative to how many sentences it has classified in total. If all sentences are classified correctly, the accuracy will have a value of 1.0.

Equation 3.12 gives the equation for accuracy.

$$\text{Accuracy} = \frac{TN + TP}{TN + TP + FN + FP} \tag{3.12}$$

Precision represents the amount of sentences correctly classified as “incorrect” relative to all the sentences that are classified as “incorrect”. If all sentences classified as “incorrect” are indeed incorrect, the precision will be 1.0.

Equation 3.13 gives the equation for precision.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.13}$$

Recall describes the amount of sentences that are correctly classified as “incorrect”, proportional to the total amount of incorrect sentences. If all the incorrect sentences are correctly classified as “incorrect”, then the recall will be 1.0.

Equation 3.14 gives the equation for recall.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.14}$$

It is necessary to consider all of these metrics when assessing the performance of the machine learning models, since considering only one or two of them does not give the complete picture. For example, a model with a low recall, which means that hardly any of the incorrect sentences are being classified incorrectly, may still reflect a high accuracy and precision, due to the different variables being compared in each calculation.

3.6. Discussion

To summarise, the first step in the design process is to devise a dataset on which to test the efficacy of the different machine learning models used in this project. Using agents instead of actual children has a variety of advantages, including that it is much easier and less time-consuming to use data from agents than to collect and use data from children. The following chapter, Chapter 4, will cover exactly how the agents are constructed from children's reading books.

The second step is to decide how to convert the agents into feature vectors, to use as input for the machine learning models. This is done with the bag-of-words method, which considers which words are present in each sentence, but discards the ordinal properties of the words, and allows similar sentences to be easily recognised by the machine learning models. Chapter 4 will describe in detail how the feature vectors are created, and will elaborate on the labelling system that is used for the vectors.

The third step is to choose machine learning models to implement and compare. The chosen models are decision trees, logistic regression, k-nearest neighbours, and neural networks. These machine learning models will be evaluated in the validation and testing phases using the precision, accuracy and recall of the models. The implementation of the models, as well as validation strategies, will be discussed in Chapter 4.

Decision trees have been chosen because they are straightforward and intuitive, consisting of a set of rules for classifying data items. Binary logistic regression, the second model chosen, is an algorithm that maximises the likelihood that a data item belongs to a particular class, by drawing a decision boundary between the classes that achieves maximal class separation. The k-nearest neighbours algorithm, which is an instance-based learning algorithm, is chosen because it inherently detects similarity between data items, in the form of calculating distances between the data items and identifying the items that are the nearest to each other. The final model is the multi-layer perceptron (MLP) model, which is a type of artificial neural network. This is the most complex model of the four, and performs its classification task with the help of hidden layers of neurons between the input layer and output layer of the model.

Chapter 4 will discuss strategies for preventing the trees from becoming excessively large, as well as the solver that is selected for use in the logistic regression algorithm and the MLP algorithm.

The fourth step is to select a method for generating new sentences from the sentences classified as "incorrect" by the two best-performing machine learning models. A context-free grammar is chosen for this purpose, as it ensures some level of grammatical correctness for the generated sentences, where the alternative that was considered, a Markov chain, has no guarantees of semantic, syntactic or grammatical correctness. Chapter 4 will detail the development of the context-free grammar.

Chapter 4

System implementation

The system is entirely software-based, written in the Python programming language [16]. Each agent is in itself an independent system. The machine learning models are trained separately on unique feature vectors for each agent, and different sentences are generated for each agent, depending on their individual sentences.

4.1. Agents

In the interest of simulating a reading child as closely as possible, it is essential to have age-appropriate reading material that is also the correct level of complexity for a first-time reader. Therefore, a set of three children's books written as a tool for teaching children to read [17] is used as the base for the agents.

To construct the agents, all the sentences that consist of three or more words and have a sensible meaning out of context, are extracted from the books. This amounts to 1 914 sentences in total. In the three books, the sentences progress from very linguistically simple in the beginning of the first book, to much more complex by the end of the third book. These sentences are divided between 19 agents, so that each agent consists of approximately the same amount of very simple and progressively more complex sentences respectively. Exactly 100 sentences are assigned to each agent.

Next, we determine the "reading errors" that each agent makes, and label each sentence for each agent according to whether it is likely to be read incorrectly or correctly by a child that makes those specific reading errors. We assume that each child, or, in this case, agent, will make a few errors consisting of either words, or individual or grouped letters (graphemes) that they consistently struggle with. Whether this is actually representative of how a child makes errors while reading is not within the scope of this project. Nevertheless, this is where [6] is useful. In [6], a table details positive rules for the top 50 miscues, in graphophonemic contexts, that were identified by the study on actual children. Many of these positive rules are used as reading errors for the agents, along with other, similar, arbitrarily devised rules corresponding to words or graphemes that the child struggles with.

The frequency with which the errors can possibly occur for a particular agent is taken into account when assigning errors to the agents. For example, an agent can only be said to "struggle" with words ending in the letter "y" if words fitting that description occur three times at the very least in the agent's sentences, and preferably more than three times. There are two reasons that this requirement is implemented when assigning errors. First, an error cannot be considered to be "common" for an agent if it occurs with a very low frequency. Second, it is unlikely that a machine learning model will even be able to recognise such an error if it occurs so few times, especially given the unusually small sizes of the datasets used in this project.

Table 4.1 lists the errors made by each agent. In the table, "%" represents a wildcard letter, that is, any one of the 26 letters of the alphabet are allowed to occur in that position. In Row 1 of the table, the word "lemma" refers to the canonical, citation form of a word – that is, the form which would be found in a dictionary. For example, "say" is the lemma of, among others, "says" and "said".

To determine which sentences contain errors for each agent, the spaCy Matcher is used [18], in conjunction with the small English language model, `en_core_web_sm`. The Matcher is a rule-matching engine that detects if patterns occur in text, and `en_core_web_sm` is a pretrained statistical model for English [18]. These patterns can include words, phrases, letter combinations or punctuation. For this project, the patterns are primarily regular expressions, or regex, which specify search patterns in character strings. A regex is useful to specify that the Matcher should only match patterns that occur at the beginning of a word, or the end of a word. If the position in the word is not specified, the Matcher will match patterns that occur anywhere in the word – beginning, end or middle.

An example of a pattern using a regex is:

```
pattern = ["LOWER": {"REGEX": "es$"}, "OP": "+"]
```

This pattern specifies that the Matcher should match any word whose lowercase form (`"LOWER":`) ends in (`$`) the letters "es", occurring one or more times (`"OP": "+"`).

Once we have identified which sentences could contain errors for each agent, using the Matcher, we take into account that a child would most likely not make the error every single time they read the words or letters they struggle with. Instead, they would possibly read them correctly in a small number of cases. To reflect this, the Matcher first identifies whether a sentence could contain any one or more of an agent's one to four errors, as specified in Table 4.1. Then, if it does contain any possible errors, we only mark it as "incorrect" a certain percentage of the time, and the remainder of the time it is marked as "correct". The percentage of the time that it was marked correct or incorrect is pre-specified, and is also used to vary the amount of errors from agent to agent, to reflect the fact that children have different levels of reading proficiency.

Agents 1 and 10–19 make their errors 90% of the time, Agents 2–4 and 6–9 make theirs

Table 4.1: Each agent makes between one and more errors, corresponding to specific words or graphemes.

Agent	Error 1	Error 2	Error 3	Error 4
1	The lemma “say”	–	–	–
2	Words ending in “es”	–	–	–
3	Words containing “ma”	Words beginning with “po”	–	–
4	Words containing “co”	Words containing “to”	–	–
5	Words containing “th”	Words ending in “ts”	Words ending in “ere”	–
6	Words ending in “o%e”	Words containing “ds”	Words containing “me”	–
7	Words containing “a”	–	–	–
8	Words containing “il”	Words containing “in”	Words containing “it”	–
9	Words ending in “e”	–	–	–
10	Words containing “p”	Words ending in “ke”	–	–
11	Words containing “he”	The word “you”	–	–
12	Words containing “of”	Words containing “x”	Words containing “as”	–
13	Words containing “all”	Words containing “oul”	Words containing “ck”	–
14	Words containing “a%e”	Words containing “u”	Words ending in “ll”	–
15	Words containing “ee”	Words containing “y”	–	–
16	Words containing “pi”	Words containing “z”	Words containing “ma”	Words beginning with “f”
17	Words containing “v”	Words containing “oo”	Words containing “ss”	–
18	Words ending in “t”	Words containing “la”	–	–
19	Words containing “ip”	Words ending in “y”	Words containing “sh”	Words ending in “st”

80% of the time, and Agent 5 makes theirs 74% of the time. This results in a minimum of seven incorrect sentences per agent, and a maximum of 71, with the average number of incorrect sentences per agent being 35. If no possible errors are identified in a sentence, it is automatically marked as “correct”.

4.2. Feature vectors

As mentioned in Chapter 3.2, we use a bag-of-words method to create feature vectors, which will be used as input for the machine learning models. To explain how this method is implemented, a simple example follows.

Let us imagine that our entire dataset consists of only the two sentences below.

```
1.  She has a bun and a bit of ham.
2.  Sal and her dad will make a pen like the pens men made in the past.
```

The bag of words for these sentences would consist of all the individual words in both of the sentences. For this project, words are case sensitive, and punctuation is included.

```
largest_possible_bagofwords = [., a, and, bit, bun, dad, ham, has, her,
  in, like, made, make, men, of, past, pen, pens, Sal, She, the, will]
```

To create a feature vector for Sentence 1 from this bag of words, we check each of the items present in the bag of words. If the item occurs in Sentence 1, the feature vector will contain a `1` at that index. If the item does not occur in Sentence 1, the feature vector will contain a `0` at that index.

The feature vectors that result for each sentence in this example are shown below. Note that each of the feature vectors has exactly the same length as the bag of words.

```
sentence1 = [1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
                                0, 0]
sentence2 = [1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0,
                                1, 1]
```

From these feature vectors, it is clear that the two sentences have the items at indices 0, 1, and 2 in common. If we reference these in the bag of words, we see that both Sentence 1 and Sentence 2 contain the items “.”, “a”, and “and”. These feature vectors do not provide any information about the order of the words in the sentence, however.

In this project, our dataset consists of not two but 1 914 sentences. A bag-of-words is then constructed from all of these sentences, and feature vectors are created for each sentence of each agent, in the same way as the feature vectors are created in this example. This results in feature vectors with a dimensionality of 1 013, meaning that there are 1 013 “words” in our bag-of-words. Although the feature vectors contain features from all 1 914 sentences, the machine learning models are trained separately on feature vectors from each individual agent.

Once the feature vectors are created for each agent, we split them into a training set and a test set. The first 85 sentences of an agent are its training set, and the remaining 15 are its test set.

Finally, we label the feature vectors according to whether they have been read correctly or incorrectly. Sentences that have been read incorrectly are labelled with a “1” and sentences that have been read correctly are labelled with a “0”. As discussed in Chapter 3.5, this means that “incorrect” is the “positive” classification and “correct” is the “negative” classification. This labelling system corresponds to the way we wish “true positives”, “false positives”, “true negatives” and “false negatives” to be defined. Therefore, it also provides a more accurate reflection of the performance of the models – we are, after all, testing whether the models can recognise which sentences would be read incorrectly by the agent.

4.3. Models

To implement the four different models used in this project – decision trees, logistic regression, k-nearest neighbours and neural networks – we make use of the scikit-learn API [19, 20]. The models are trained on each agent’s training feature vectors, which have been described in the previous section. The input data for the models for each agent consists of the training feature vectors of that agent, X, along with the labels of those vectors, y. The output data is the predicted labels for each set of feature vectors, y_{pred}. Unless otherwise specified, default parameters are used for all the models.

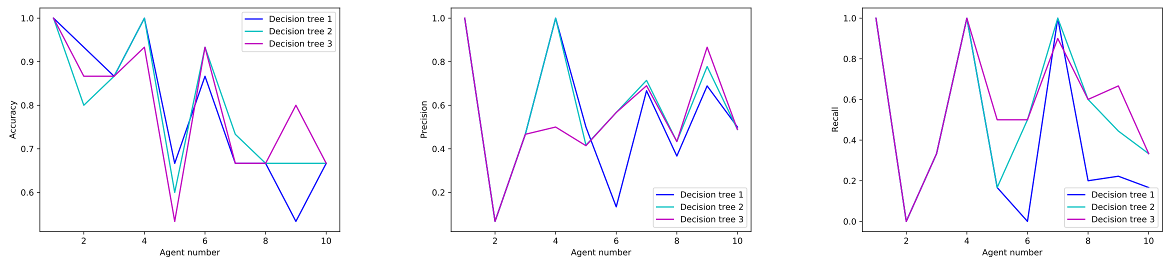
For the logistic regression models and the neural networks, we use the “LBFGS” solver, an optimisation algorithm that implements a quasi-Newton method. This solver, according to [19], performs the best on smaller datasets, which is what we are using in this project.

The validation strategy we use is to implement three different versions of each model, each with different hyperparameters that contribute to the learning process. The same three versions are implemented separately for Agents 1–10, and the test data, X_{test}, with their labels y_{test}, are used to calculate the accuracy, precision, and recall for each of those agents. These parameters are then compared to find the best of the three sets of hyperparameters for each model.

4.3.1. Decision trees

The hyperparameters for the decision trees are used to control the tree size. A larger tree could be more accurate, due to increased leaf purity, or it could become very specialised and therefore less accurate. Bearing this in mind, the hyperparameters for the three decision tree models are the maximum number of leaf nodes and the minimum impurity decrease.

The maximum number of leaf nodes restricts the overall size of the tree, and the



(a) **(b)** **(c)**

Figure 4.1: (a) The accuracy values for the ten validation agents are measured and compared for each of the three decision tree models. (b) The precision values for the ten validation agents are measured and compared for each of the three decision tree models. (c) The recall values for the ten validation agents are measured and compared for each of the three decision tree models.

minimum impurity decrease governs when a node will be split into two new nodes. If the maximum number of leaf nodes is increased, the tree will be allowed to grow larger. For a smaller minimum impurity decrease, the tree may also be allowed to grow larger – there will be more leaf nodes. Therefore, during validation of the decision trees, it would be rather pointless to decrease both the maximum number of leaf nodes and the minimum impurity decrease, or increase both, as they would almost certainly counteract each other.

Therefore, for the three different decision tree validation models, when the maximum number of leaf nodes hyperparameter is increased, then the miminum impurity decrease hyperparameter is decreased. For the first model, Decision Tree One, the maximum number of leaf nodes is `max_leaf_nodes = 10`, and the minimum impurity decrease is `min_impurity_decrease = 0.05`. For the second model, Decision Tree Two, the maximum number of leaf nodes is `max_leaf_nodes = 15`, and the minimum impurity decrease is `min_impurity_decrease = 0.025`. For the third model, Decision Tree Three, the maximum number of leaf nodes is `max_leaf_nodes = 20`, and the minimum impurity decrease is `min_impurity_decrease = 0.01`. The intended result of these parameter values is that the size of the tree is restricted the most for Decision Tree One, and the least for Decision Tree Three.

Figures 4.1a, 4.1b and 4.1c show the accuracy, precision and recall of the three different models respectively. It is clear from these figures that the precision, accuracy and recall vary significantly from agent to agent. However, the precision, recall and accuracy values for each agent are in most cases quite similar to each other, and as such there is not one decision tree model that clearly performs better than the other two.

To gain more insight into the performance of the three decision tree models, the average accuracy, average precision and average recall are calculated for each model. The results are shown in Table 4.2. From the values in the table, it is clear that, on average, both Decision Tree Two and Decision Tree Three perform better in all aspects than Decision Tree One. Decision Trees Two and Three have the same accuracy, Decision Tree Two has

Table 4.2: The average accuracy, average precision and average recall is calculated for each of the decision tree validation models. The best average accuracy, average precision and average recall values are shown in bold.

Decision tree model	Average accuracy	Average precision	Average recall
1	0.7867	0.5389	0.4089
2	**0.7933**	**0.5931**	0.5378
3	**0.7933**	0.5493	**0.5833**

the best precision and Decision Tree Three has the best recall. To determine which of these two models is the best, we return to our definitions of precision and recall. In Chapter 3.5, we define precision as the amount of sentences correctly classified as "incorrect" relative to all the sentences that are classified as "incorrect". In other words, it measures how many of the sentences that the model classifies as incorrect, are in fact incorrect. Recall, on the other hand, is defined as the amount of sentences that are correctly classified as "incorrect", proportional to the total amount of incorrect sentences. This means that it measures how many of the incorrect sentences it correctly classifies as incorrect.

For this project, one of our aims, as described in the problem statement in Chapter 1.1, is to implement a machine learning model that identifies the types of sentences that a first-time reader consistently struggles with. This means that we want to implement a model that can correctly identify which sentences have been read incorrectly by the reader. The definition of recall aligns much better with this aim than the definition of precision. Therefore, we choose Decision Tree Three as our decision tree model, since it has the highest values of the three for accuracy and recall.

To further confirm this, it is useful to examine one of the decision tree models and see how the accuracy, precision and recall are calculated. Let us examine Decision Tree Two for Agent 7. The decision tree that is grown is shown in Figure 4.2.

This decision tree is making decisions based on five features: `X[558]`, `X[579]`, `X[898]`, `X[216]`, and `X[288]`. These correspond to the words "just", "left", "tent", "where" and

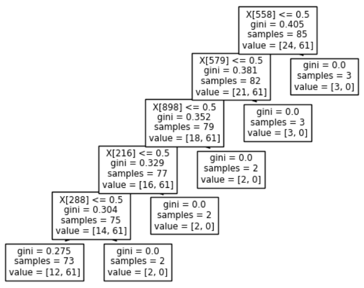


Figure 4.2: Decision Tree Two for Agent 7 is one of the decision trees grown for this project.

Table 4.3: The true labels depend on whether or not words with “a” are present in the sentences, with an 80% probability of being labelled “incorrect” if these kinds of words are present. The predicted labels depend on whether the words “just”, “left”, “tent”, “where” and “bit” are present in the sentence, and the label is predicted as “incorrect” if at least one of them is present.

Sentence	Error present?	Feature	True label	Predicted label
1	Yes	–	1	1
2	Yes	–	1	1
3	Yes	–	1	1
4	Yes	bit	0	0
5	Yes	–	0	1
6	Yes	–	1	1
7	Yes	–	1	1
8	Yes	–	1	1
9	Yes	–	1	1
10	Yes	–	1	1
11	No	–	0	1
12	Yes	–	0	1
13	Yes	–	1	1
14	No	–	0	1
15	Yes	–	1	1

“bit” respectively in the bag of words from which the feature vectors were constructed. The model is making the prediction that if any sentence contains one of those five features, it will be read correctly by the agent. Incidentally, this does not contradict the error made by the agent, which is that it struggles with words containing the letter “a” (see Table 4.1), since none of the words listed above contain an “a”.

Table 4.3 compares the actual and predicted labels of the sentences. From this table, we can identify the amounts of true negatives (TN), true positives (TP), false negatives (FN), and false positives (FP), as defined in Chapter 3.5. From these four values we can calculate the accuracy, precision and recall of this model, according to the equations in Chapter 3.5.

We can see that $TN = 1$, $TP = 10$, $FN = 0$, and $FP = 4$. This yields an accuracy of 0.7333, a precision of 0.7143 and a recall of 1.0, as shown in Figures 4.1a, 4.1b, and 4.1c. We can see that all of the sentences that are actually incorrect, have, in fact, been classified as incorrect, as shown by the perfect recall score. Some of the sentences that are actually correct, however, have been classified as incorrect, as shown by the imperfect precision score. For this project, it is more desirable to have all the incorrect sentences recognised than to have all the correct sentences recognised. This confirms that recall is a better metric to prioritise than precision.

4.3.2. Logistic regression

The hyperparameters used for the three logistic regression models are C, the regularisation parameter, and the maximum number of iterations. C is the inverse of regularisation strength, as explained in Chapter 3.3.2, so a high value for C amounts to minimal or no regularisation and a low value for C amounts to stronger regularisation. The maximum number of iterations limits how many times the solver may iterate when attempting to achieve convergence.

Stronger regularisation can provide greater numerical stability, but it may or may not improve accuracy, as it prevents specialisation. The maximum number of iterations directly affects training time, and, again, may or may not affect accuracy, especially if the model is not allowed enough iterations to converge sufficiently. When validating these hyperparameters, the maximum number of iterations is increased as the regularisation parameter C is decreased. This means that progressively stronger regularisation will be applied to the model, and at the same time it will be allowed more iterations to achieve convergence.

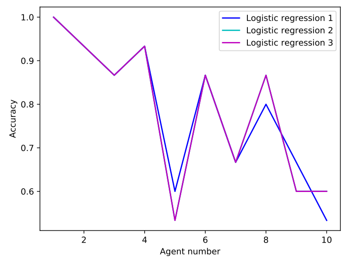
For the first logistic regression validation model, known as Logistic Regresson One, the regularisation parameter is $C = 100\,000$ and the maximum number of iterations is 100. For the second model, known as Logistic Regression Two, the regularisation parameter is $C = 10000$ and the maximum number of iterations is 300. For the third model, Logistic Regression Three, the regularisation parameter is $C = 1000$ and the maximum number of iterations is 500.

Figures 4.3a, 4.3b, and 4.3c show the accuracy, precision and recall respectively of these models for each agent. As with the decision trees, the values for accuracy, precision and recall are very similar for each agent, but vary substantially between agents.

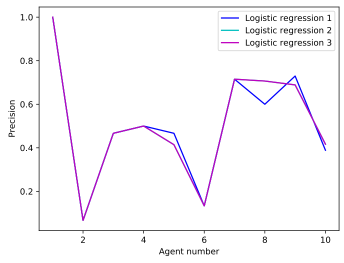
Table 4.4 shows the average accuracy, average precision and average recall for the three validation models. We can see that Logistic Regression One, Two and Three all have the same average accuracy, but Logistic Regression One has the highest average recall and Logistic Regression Two and Three have the highest average precision. As we discussed in the previous section on decision trees, we prioritise recall for this project, and therefore we choose Logistic Regression One as our best model.

Table 4.4: The average accuracy, average precision and average recall is calculated for each of the logistic regression validation models. The best average accuracy, average precision and average recall values are shown in bold.

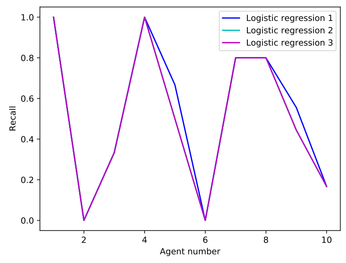
Logistic regression model	Average accuracy	Average precision	Average recall
1	**0.7867**	0.5067	**0.5322**
2	**0.7867**	**0.5108**	0.5044
3	**0.7867**	**0.5108**	0.5044



(a)



(b)



(c)

Figure 4.3: (a) The accuracy values for the ten validation agents are measured and compared for each of the three logistic regression models. (b) The precision values for the ten validation agents are measured and compared for each of the three logistic regression models. (c) The recall values for the ten validation agents are measured and compared for each of the three logistic regression models.

4.3.3. k-nearest neighbours

For k-nearest neighbours (KNN), the hyperparameters we use in validation are the number of nearest neighbours, and the method used to calculate the distance between the neighbours. We examine the effects of having more or fewer neighbours and using different distance calculations on model performance.

For KNN One, the first model, we use three neighbours and the Minkowski distance with $p = 3$ (recall Equation 3.8 from Chapter 3.3.3). For KNN Two, the second model, we use one neighbour and the Manhattan distance. Finally, for KNN Three, the third model, we use two neighbours and the Euclidean distance.

Figures 4.4a, 4.4b, and 4.4c show the accuracy, precision and recall respectively of these models for each agent. The three models have quite similar accuracy and precision for each agent, but there is more variation in recall. There is also, as with decision trees and logistic regression, a discrepancy between the accuracy, precision and recall results between agents.

Table 4.5 shows the average accuracy, average precision and average recall for each of the KNN models. KNN Three clearly performs the worst in all aspects. KNN One has slightly better average accuracy and average precision than KNN Two, but KNN Two has the best recall. As mentioned in previous sections, we prioritise recall for this project, so we choose KNN Two as our best model, especially since the difference in accuracy is not large.

4.3.4. Neural networks

All three of the neural network models have the regularisation parameter set to `alpha` = 0.00001, and for reproducibility, the `random_state` parameter, which determines the generation of random numbers for weights and bias initialisation, is set to 1.

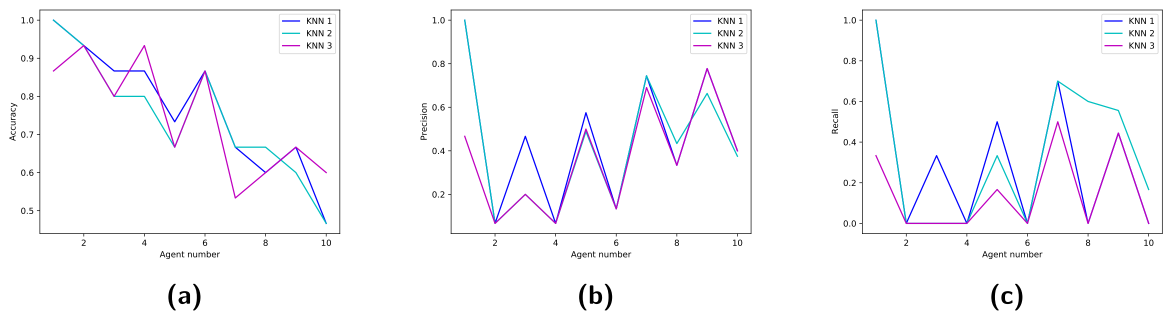


Figure 4.4: (a) The accuracy values for the ten validation agents are measured and compared for each of the three k-nearest neighbours models. (b) The precision values for the ten validation agents are measured and compared for each of the k-nearest neighbours models. (c) The recall values for the ten validation agents are measured and compared for each of the three k-nearest neighbours models.

The hyperparameter that is tuned during validation is the number and size of the hidden layers.

The first neural network (NN), NN One, has two hidden layers, of sizes $(100, 50)$. The second neural network, NN Two, also has two hidden layers, of sizes $(200, 100)$. The third neural network, NN Three, has three hidden layers, of sizes $(200, 100, 50)$.

Figures 4.5a, 4.5b, and 4.5c show the respective accuracy, precision, and recall of these models, for each of the ten validation agents. Again, the variation of values between agents is large, although this time the values per agent differ quite significantly in some cases.

Table 4.6 shows the average accuracy, average precision and average recall of each of the neural network models on the development data. NN Two has the best average accuracy and precision, while NN One has the best average recall. For the other three machine learning models, we chose the model with the best average recall in each case. However, for the neural networks, the differences between the average accuracy and average precision for NN One and NN Two are too large to justify such a decision. Therefore, we choose NN Two as our best neural network model.

Table 4.5: The average accuracy, average precision and average recall is calculated for each of the k-nearest neighbours validation models. The best average accuracy, average precision and average recall values are shown in bold.

KNN model	Average accuracy	Average precision	Average recall
1	**0.7667**	**0.4564**	0.2978
2	0.7467	0.4172	**0.3356**
3	0.7467	0.3635	0.1444

Table 4.6: The average accuracy, average precision and average recall is calculated for each of the neural network validation models. The best average accuracy, average precision and average recall values are shown in bold.

Neural network	Average accuracy	Average precision	Average recall
1	0.7600	0.5319	**0.6522**
2	**0.7867**	**0.5979**	0.6489
3	0.7800	0.5947	0.6356

4.4. Sentence generation

As discussed in Chapter 3.4, a context-free grammar is used in this project, in order to generate similar sentences to those each agent struggles with. To accomplish this, the `CFG` class from the `grammar` module from NLTK [21] was used to construct context-free grammars, and the `nltk.parse.generate` module was used to generate sentences from those context-free grammars.

To better understand how a context-free grammar is constructed, Figure 4.6 shows the tree structure obtained from the sentence “The man looks at the box.” From this tree structure, we can deduce a set of recursive productions for generating sentences with a similar structure. These productions can be used to form a context-free grammar, which would look like the following:

```
S  -> NP VP
NP -> Det Nom
Nom -> N
VP -> V PP
PP -> P NP
Det -> ‘the’
```

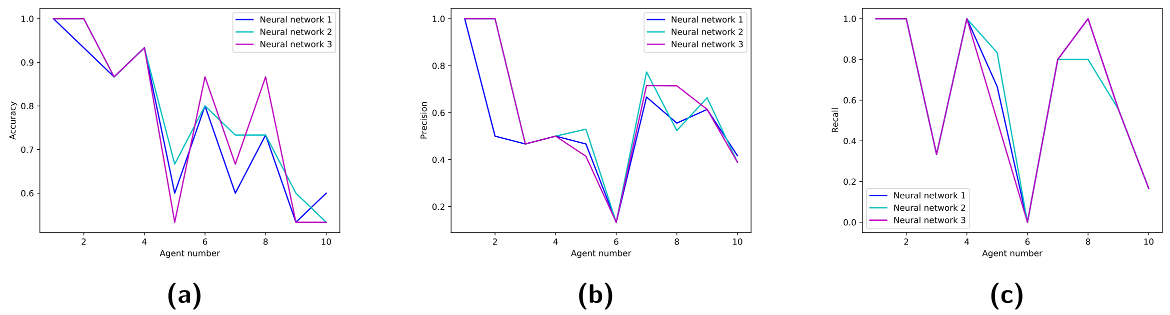


Figure 4.5: (a) The accuracy values for the ten validation agents are measured and compared for each of the three neural network models. (b) The precision values for the ten validation agents are measured and compared for each of the three neural network models. (c) The recall values for the ten validation agents are measured and compared for each of the three neural network neighbours models.

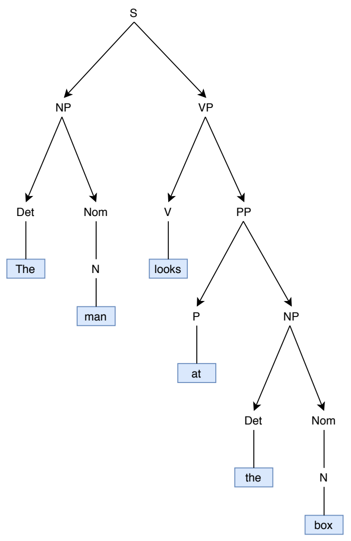


Figure 4.6: The sentence “The man looks at the box.” is parsed into a recursive context-free grammar that contains nested nominal phrases. This is an example of how a context-free grammar is created.

```
N -> ‘man’ | ‘box’
V ->  ‘looks’
P -> ‘at’
```

The left-hand side of the first production `S -> NP VP` is the start-symbol of the grammar, according to convention, and is usually called `S`. This is the root label of all trees constructed from this grammar. It tells us that a sentence consists of a noun phrase (`NP`) followed by a verb phrase (`VP`). The second production tells us that a noun phrase consists of a determinant (`Det`) followed by a nominal (`Nom`), which the third production tells us is really just a noun (`N`). The fourth production tells us that a verb phrase consists of a verb (`V`) followed by a prepositional phrase (`PP`), and the fifth production tells us that a prepositional phrase is a preposition (`P`) followed by a noun phrase. This grammar is said to be recursive because categories occurring on the left-hand side of productions also occur on the right-hand side of (other) productions.

Exactly four sentences can be generated from this grammar:

```
the man looks at the man
the man looks at the box
the box looks at the man
the box looks at the box
```

The grammars used in this project are derived from one of the example grammars from NLTK. All the grammars contain the following productions:

```
S -> NP VP
PP -> P NP
```

```
NP -> Det N | NP PP
VP -> V NP | VP PP
```

The remaining productions, which specify the words from which sentences are to be constructed by the grammar, are derived from the sentences of Agents 11–19 that are classified as incorrect by either NN Two or Decision Tree Three, our best neural network and decision tree. Agents 11–19 are our testing agents, and NN Two and Decision Tree Three are our two best-performing models (compared also to KNN Two and Logistic Regression One, the best KNN and logistic regression models). Therefore, two different context-free grammars are constructed for each of the testing agents, one for each machine learning model, amounting to 18 context-free grammars in total.

To derive these remaining productions, all the nouns, verbs, prepositions and determinants are extracted from the sentences classified as incorrect by NN Two or Decision Tree Three respectively. These are then added to a context-free grammar string that already contains the productions that are common to all 18 grammars. These 18 context-free grammars are then used to generate 18 unique sets of sentences.

4.5. Discussion

Implementation of the project begins with creating agents that simulate a reading child as closely as possible. To accomplish this, we use children's reading books to create text-based agents that each make a set of one to four, usually graphophonemic, "reading errors".

Using a bag-of-words method, feature vectors are extracted from the words of each agent, to allow machine learning models to identify sentences in each agent that contain similar words. Sentences read incorrectly are labelled with a "1" and sentences read correctly are labelled with a "0".

Three validation models are implemented to tune the decision tree hyperparameters, and likewise for logistic regression, k-nearest neighbours, and neural networks. Decision Tree Three, the third validation model, is chosen as the best decision tree model, since it has the best average recall, which we generally prioritise in this project. Logistic Regression One is chosen as the best logistic regression model, as it has the best average recall. KNN Two is chosen as the best k-nearest neighbours model, since it has the highest average recall. NN Two is chosen as the best neural network model, since it has a higher accuracy and precision than the other two models, by a sizeable margin for both, and its recall is not significantly worse. Chapter 5 will describe how we evaluate the best model between Decision Tree Three, Logistic Regression One, KNN Two and NN Two.

We construct 18 different context-free grammars for sentence generation. Each has a set of productions that are common to all 18 grammars, and a unique set of words with which to construct sentences based off the common production rules. These words are extracted from the sentences classified as incorrect by either NN Two or Decision Tree Three. Chapter 5 will detail the evaluation of the context-free grammars, to gauge their efficacy in generating sentences that are similar to those that are actually read incorrectly by each agent. Additionally, Chapter 5 will compare the context-free grammars generated for each agent from the two different machine learning models, to judge which model generates more appropriate sentences.

Chapter 5

System evaluation

Following the implementation and validation of the decision tree, logistic regression, k-nearest neighbours and neural network models, it is necessary to assess which models perform the best overall. Once we have chosen the two best models, we use these to construct the context-free grammars, as explained in Chapter 4.4. With regard to these grammars, we test whether the sentences that they are used to generate are in fact similar to those the agents struggle with. This is also a method by which we can test the performance of the two best models – if one of the best models results in more relevant sentences being generated than the other, then that is the preferred model.

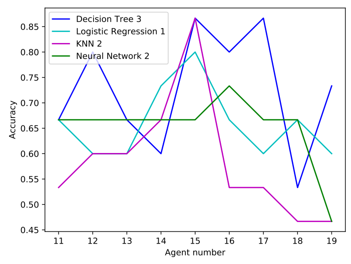
5.1. Analysis of model performance

We evaluate the performance of the decision trees, logistic regression, k-nearest neighbours and neural networks using accuracy, precision and recall. These metrics are obtained for each by testing the best decision tree, logistic regression, k-nearest neighbours and neural network models, as selected in Chapter 4, on Agents 11–19, the nine testing agents.

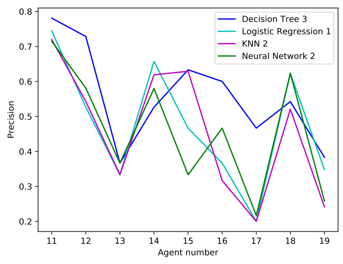
5.1.1. Selecting the best model

Figures 5.1a, 5.1b, and 5.1c show the per-model accuracy, precision, and recall of Decision Tree Three, Logistic Regression One, k-Nearest Neighbours (KNN) Two and Neural Network (NN) Two. It is clear from Figure 5.1a that Decision Tree Three often displays a higher level of accuracy than the other models, and from Figure 5.1b that Decision Tree Three also often has a higher level of precision than the other models. Figure 5.1c shows that NN Two often has the highest recall. However, in all cases there is a large amount of variation between agents.

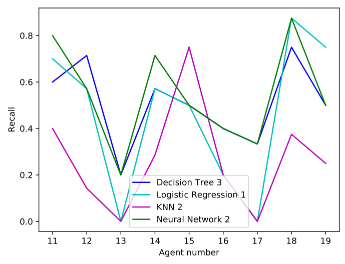
Table 5.1 shows the average accuracy, average precision and average recall as calculated for Decision Tree Three, Logistic Regression One, KNN Two and NN Two. From this table, we can see that Decision Tree Three has the best average accuracy, followed by Logistic Regression One, NN Two and finally KNN Two. Decision Tree Three also has the best average precision, followed again by Logistic Regression One, NN Two and KNN



(a)



(b)



(c)

Figure 5.1: (a) The accuracy values for the nine testing agents are measured and compared for the decision tree, logistic regression, k-nearest neighbours and neural network models. (b) The precision values for the nine testing agents are measured and compared for the decision tree, logistic regression, k-nearest neighbours and neural network models. (c) The recall values for the nine testing agents are measured and compared for the decision tree, logistic regression, k-nearest neighbours and neural network models.

Two. NN Two, however, has the best recall, followed by Decision Tree Three, Logistic Regression One, and lastly KNN Two.

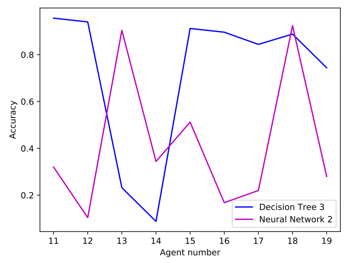
Since it performs the best in average accuracy and average precision, we choose Decision Tree Three as our best model. NN Two is chosen as our second-best model, since it has the best recall, which is important for this project, as discussed in Chapter 4.

Table 5.1: The average accuracy, average precision and average recall is calculated for Decision Tree Three, Logistic Regression One, k-Nearest Neighours Two and Neural Network Two. The best average accuracy, average precision and average recall values are shown in bold.

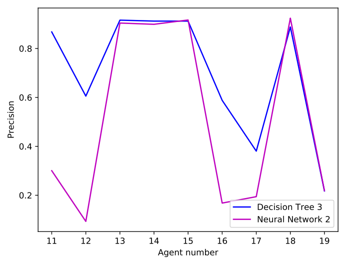
Model	Average accuracy	Average precision	Average recall
Decision Tree Three	**0.7260**	**0.5587**	0.5077
Logistic Regression One	0.6593	0.4740	0.4631
KNN Two	0.5852	0.4582	0.2671
NN Two	0.6519	0.4601	**0.5438**

5.2. Context-free grammar

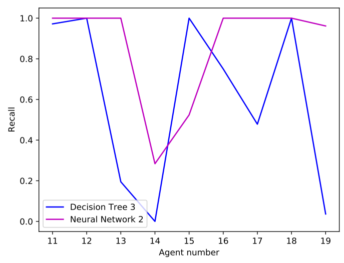
The objective behind the implementation of the contex-free grammar is to generate sentences for each agent that are similar to those the agent reads incorrectly. To determine whether this is being achieved, we test two aspects of the context-free grammar. First, we predict separately for Neural Network (NN) Two and Decision Tree Three the proportion of generated sentences that contain errors for an agent. Second, we test the accuracy, precision and recall of that prediction and compare it to the actual proportion of generated sentences that contain errors for that agent. To test these values, we randomly select 250



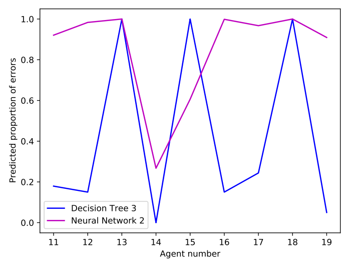
(a)



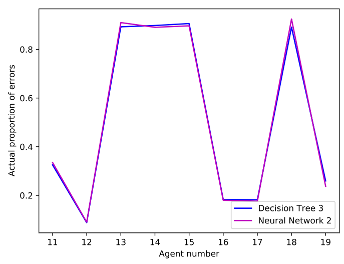
(b)



(c)



(d)



(e)

Figure 5.2: (a) The accuracy values for the sentences generated for the nine testing agents are measured and compared for the decision tree and neural network models. (b) The precision values for the sentences generated for the nine testing agents are measured and compared for the decision tree and neural network models. (c) The recall values for the sentences generated for the nine testing agents are measured and compared for the decision tree and neural network models. (d) The predicted proportion of sentences for each of the agents that contain errors is measured and compared for the decision tree and neural network models. (e) The actual proportion of sentences for each of the agents that contain errors is measured and compared for the decision tree and neural network models, and is found to be nearly identical for both models.

of the sentences generated by each model, for each agent, and predict labels for them using the previously-trained models.

Figure 5.2d shows the predicted proportion of generated sentences for each agent that contain errors. Figures 5.2a, 5.2b and 5.2c show the accuracy, precision and recall of this prediction, and Figure 5.2e shows the actual proportion of generated sentences for each agent that contain errors.

It is clear from Figures 5.2a and Figure 5.2b that Decision Tree Three has higher levels of accuracy and precision in the majority of cases, as our previous test results on the original agent sentences suggested. From Figure 5.2c we can see that NN Two has a higher, and also near perfect, recall in most cases. The reason for this becomes evident when we look at Figure 5.2d, and see that NN Two predicts that a large proportion of the generated sentences contain errors – if the model predicts that most of the sentences contain errors, the recall will likely be high as the actual incorrect sentences will probably be included in the many sentences that are predicted as incorrect, although the precision will therefore also likely be low in these cases. Again, this aligns with our previous test results.

Figure 5.2e shows an interesting result – the true proportion of errors that occur in the sentences generated by each model, for each agent, is nearly identical in every single case. This strongly suggests that the proportion of errors in each set of generated sentences is solely dependent on the individual agents, and that the models do not have any effect on how many generated sentences contain errors. This is contrary to what was expected, namely that the predictions made by the model would strongly affect the generation of the sentences, which would then have an effect on the proportion of errors in the sentences.

Table 5.2 shows the average accuracy, average precision, average recall, predicted average proportion of sentences containing errors, and actual average proportion of sentences containing errors. The values in this table are consistent with the previous test results, on the original agent sentences, in that NN Two still has the highest average recall and Decision Tree Three still has the highest average accuracy and average precision. However, these values, measured on a larger test set of 250 sentences, are slightly different from those measured on a smaller test set of 15 sentences. In some cases the values even vary by as much as 25%, in the case of the average precision of Decision Tree Three, or even 58%, in the case of the average recall of NN Two, which is truly a significant variation. This suggests that the size of the test set may have an effect on the evaluation of the models.

The table also confirms that the recall of NN Two is very high because it predicted a large proportion of the sentences as containing errors – there is a very large (0.3357) discrepancy between the actual and predicted proportion of sentences containing errors for NN Two. The discrepancy is much smaller for Decision Tree Three (only 0.0963). This presumably plays a role in the higher accuracy and precision of Decision Tree Three.

Table 5.2: The average accuracy, average precision, average recall, predicted average proportion of sentences containing errors and actual average proportion of sentences containing errors is compared for Neural Network Two and Decision Tree Three. The best average accuracy, average precision and average recall values are shown in bold.

Metric	Neural Network Two	Decision Tree Three
Average accuracy	0.4196	**0.7222**
Average precision	0.5131	**0.6985**
Average recall	**0.8632**	0.6035
Predicted proportion of sentences containing errors	0.8503	0.4193
Actual proportion of sentences containing errors	0.5156	0.5139

5.3. Discussion

When we analyse the performance of the best decision tree, logistic regression, k-nearest neighbours and neural network models respectively, we find that Decision Tree Three has the best average accuracy and precision, and this is chosen as our overall best model. Neural Network (NN) Two has the best recall, however, so this model is chosen as our second-best model.

To assess how similar the sentences generated with the help of the context-free grammar are to those read incorrectly by the agents, we use NN Two and Decision Tree Three, the two best models, to predict the proportion of a set of sentences generated from each model that contain errors. We find that NN Two predicts that most of the sentences, around 80%, contain errors, on average, which contributes to the high recall values displayed by the model. Decision Tree Three, on the other hand, predicts that only around 40% of the sentences contain errors, on average. In reality, the proportion of sentences for each model that contain errors is on average approximately 50% for both. Indeed, it is nearly identical for both models, on average and when compared for each agent. This means that both models generate sentences that are equally relevant to the agents' individual errors.

Chapter 6 summarises our findings and draws conclusions based on the results.

Chapter 6

Summary and Conclusion

Machine learning has a great role to play in improving the low literacy rates in South Africa, in part due to the personalisation it can contribute to a first-time reader's learning experience. Although ideal, traditional machine learning models that are trained on large corpora are not a practical choice for this application, not least because most South African languages are low-resource in nature and large quantities of data are simply not available in these languages.

6.1. Machine learning for improving reading literacy

This project has explored an alternative to the aforementioned traditional methods of training machine learning models, by investigating the viability of training machine learning models on much smaller datasets, with the objective of identifying sentences that an early reader consistently struggles with. The predictions made by these models can then be used to generate similar sentences to those the reader struggles with, so that they can practise reading challenging sentences and improve. In this way, the advantage of personalisation that machine learning can bring is still included in the reader's education.

To accomplish this, we first constructed 19 different agents, in an attempt to simulate children who read a sentence either correctly or incorrectly. For this purpose, we used children's reading books to create text-based agents that each make between one and four, usually graphophonemic, "reading errors", as inspired by [6]. These agents were then converted into feature vectors, using a bag-of-words method, to allow machine learning models to identify sentences in each agent that contain similar words. Sentences read incorrectly are labelled with a "1", while sentences read correctly are labelled with a "0".

Next, we compared different machine learning models to assess which perform best when trained and tested on small datasets. The four different types of machine learning models we implemented were decision trees, logistic regression, k-nearest neighbours and neural networks. To validate our models, we compared models trained using three different sets of hyperparameters, and selected the model that performed the best in terms of accuracy, precision and recall. We then analysed the performance of the best decision tree, logistic regression, k-nearest neighbours and neural network models respectively. We found that our decision tree model had the best average accuracy (0.7260) and precision (0.5587), and a recall of 0.5077, so we chose this as our overall best model. Our neural network, which had an accuracy of 0.6519 and a precision of 0.4601, had the best recall (0.5438), however, so we chose this model as our second-best model. We decided that recall is an important metric for this project, as it represents how many of the incorrectly read sentences are correctly recognised as incorrect, which is in essence what our agents are trying to achieve.

To realise our final goal, which was to generate sentences that are relevant to the specific issues a reader experiences while reading, we implemented context-free grammars that were constructed from the sentences for each agent that were predicted as incorrect by our best neural network and decision tree models. These context-free grammars were then used to generate new sentences for each agent, using NLTK's API [21]. In an attempt to confirm that this goal was indeed met, we used the previously-trained decision trees and neural networks to predict the proportion of 250 sentences generated by each model for each agent that contained the errors of that agent. We evaluated the accuracy, precision and recall of this prediction, and compared it to the actual proportion of sentences generated for each agent that contained errors. We found that the neural networks predicted a much higher proportion (an average of around 80%) of sentences that contained errors, and the decision trees predicted a rather low proportion of sentences that contained errors (an average of around 40%). As expected based on the previous tests, the recall of the neural networks was higher on average than that of the decision trees – 0.8632 as compared to 0.6035 – but the precision and accuracy of the decision trees, which amounted to an average of 0.6985 and 0.7222 respectively, were higher than those of the neural networks, which were on average 0.5131 and 0.4196 respectively.

In reality, the proportion of generated sentences that contained errors was very nearly identical for each model, with an average proportion of 0.5156 for the neural networks and 0.5139 for the decision trees. From this we can conclude that the proportion of errors is entirely agent-dependent, and is not affected by whether the context-free grammar that generated the sentences was constructed using predictions from a neural network or from a decision tree.

To conclude, the best-performing models for small training and testing datasets are, according to this project, decision trees and neural networks, with decision trees having a higher accuracy and precision, and neural networks having a higher recall. The choice of model does not affect the system's ability to generate sentences that contain errors made by each agent, so both decision trees and neural networks result in sentences that are relevant to the agent's individual reading errors around 50% of the time.

6.2. Future work and recommendations

This project was concerned with finding the most appropriate type of machine learning model to train and test on small datasets. Future work could include further investigating how to improve a decision tree or neural network, for example, to make it even better suited for smaller datasets. Alternatively, since the sentence generation system is already implemented, these generated sentences could be used to augment the datasets, in order to have a larger dataset on which to train the models.

The feature vectors that were used in this project, bag-of-words feature vectors, do not preserve any of the ordinal properties of the sentences. However, it may be possible that a child could mispronounce words only if they are in a certain order or context. Therefore, another suggestion for future work is to experiment with different feature vectors that do take the order of words in a sentence into account.

With regard to sentence generation, the context-free grammars that were used in this project were very basic – implementing an elaborate sentence generation algorithm was not within the scope of this project. A more detailed and sophisticated context-free grammar could improve the variety and grammatical, syntactical and semantic correctness of the sentences that are generated. One suggestion is to parse the sentences that were read incorrectly, to directly create context-free grammars based on the structure of those sentences, instead of merely collecting words from them as was done in this project.

In addition, since not all of the generated sentences do or will contain the errors made by the individual agents, it is recommended to use the previously-trained machine learning models to predict which of the sentences do contain errors. Although the predictions are often not wholly accurate, this is preferable to simply assuming that all the sentences contain errors.

Finally, in this project, we constructed agents and then used machine learning to essentially “reverse engineer” the type of mistakes they would make. This approach was useful for this project, which provides a good foundation for a systematic comparison of identification techniques, and also explores sentence generation. However, real experiments on children would definitely be required in future work, as real children are obviously very different from constructed agents, and their reading errors are much more nuanced. This would, of course, require the acquisition of special ethical clearance.

Bibliography

[1] S. Howie, C. Combrinck, K. Roux, M. Tshele, G. Mokoena, and N. M. Palane, *PIRLS Literacy 2016: Progress in International Reading Literacy Study 2016: South African children's reading literacy achievement.* Centre for Evaluation and Assessment (CEA), Faculty of Education, University of Pretoria, 2017.

[2] K. Lee, A. Hagen, N. Romanyshyn, S. Martin, and B. Pellom, "Analysis and detection of reading miscues for interactive literacy tutors," in *Proceedings of the 20th International Conference on Computational Linguistics.* Geneva, Switzerland: COLING, August 2004, pp. 1254–1260. [Online]. Available: https://www.aclweb.org/anthology/C04-1182

[3] A. S. Cohen, "Oral reading errors of first grade children taught by a code emphasis approach," *Reading Research Quarterly*, vol. 10, no. 4, pp. 616–650, 1974-1975.

[4] R.-M. Weber, "A linguistic analysis of first-grade reading errors," *Reading Research Quarterly*, vol. 5, no. 3, pp. 427–451, 1970.

[5] M. Black, J. Tepperman, S. Lee, P. Price, and S. Narayanan, "Automatic detection and classification of disfluent reading miscues in young children's speech for the purpose of assessment," in *Proceedings of the Eighth Annual Conference of the International Speech Communication Association*, Antwerp, Belgium, August 2007, pp. 206–209.

[6] J. Fogarty, L. Dabbish, D. M. Steck, and J. Mostow, "Mining a database of reading mistakes: For what should an automated reading tutor listen?" in *Proceedings of the Tenth Artificial Intelligence in Education (AI-ED) Conference*, May 2001.

[7] A. Kazemzadeh, H. You, M. Iseli, B. Jones, X. Cui, M. Heritage, P. Price, E. Andersen, S. Narayanan, and A. Alwan, "Tball data collection: the making of a young children's speech corpus." in *Proceedings of the Ninth European Conference on Speech Communication and Technology*, January 2005, pp. 1581–1584.

[8] Scikit-learn developers, "1.10. Decision Trees," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html#tree

[9] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer Series in Statistics. Springer New York, 2013. [Online]. Available: https://books.google.co.za/books?id=yPfZBwAAQBAJ

[10] B. Herbst and J. Du Preez, "Machine Learning."

[11] Scikit-learn developers, "1.1.11. Logistic regression," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

[12] ——, "1.6.2. Nearest Neighbors Classification," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification

[13] ——, "1.17. Neural network models (supervised)," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html

[14] Ozdemir, Alex, "Generating Personal Statements," 2020. [Online]. Available: https://cs.stanford.edu/~aozdemir/blog/generating-personal-statements/

[15] Goddard, Wayne, "Context-Free Grammars," 2020. [Online]. Available: https://people.cs.clemson.edu/~goddard/texts/theoryOfComputation/6a.pdf

[16] Python Software Foundation, "Python Language Reference, version 3.7.8," 2001–2020. [Online]. Available: https://www.python.org/

[17] J. Holzmann, *I Can Read It, Books 1-3.* Littleton, CO, USA: Sonlight Curriculum, Ltd., 2005.

[18] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017, to appear.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[21] S. Bird, E. Loper, and E. Klein, *Natural Language Processing with Python.* O'Reilly Media Inc, 2009.

Appendix A

Project Planning Schedule

My first meeting with my supervisor was on 30 June, 2020, which was when we outlined the project. The project goals were clarified on 6 July 2020, and work began on the project immediately thereafter, following the planning schedule detailed in Table A.1.

Table A.1: The goals of the project were set within a planned time frame.

Time frame	Planned goals
6 July 2020 – 20 July 2020	Literature study on children's reading miscues and commencement of construction of text-based agents.
20 July 2020 – 27 July 2020	Continuation of agent construction, creation of first bag-of-words feature vector, and implementation of first decision tree model.
27 July 2020 – 3 August 2020	Continuation of agent construction, creation of further feature vectors, implementation of remaining decision tree models, and commencement of model validation process.
3 August 2020 – 31 August 2020	Remainder of agents constructed, remainder of feature vectors created, implementation, validation and testing of logistic regression, k-nearest neighbours and neural network models, as well as testing of decision tree models.
31 August 2020 – 21 September 2020	Development of context-free grammars and sentence generation algorithm.
21 September 2020 – 5 October 2020	Final experiments and writing first report draft for feedback.
5 October 2020 – 30 October 2020	Complete second draft of report for feedback.
30 October 2020 – 9 November 2020	Complete final draft of report and submit.

Appendix B

Outcomes Compliance

All required ECSA exit level outcomes were achieved while executing this project and writing this report. These outcomes, along with a description of how they were achieved and which chapters of the report are relevant to each outcome, are shown in Tables B.1, B.2, and B.3.

Table B.1: Each ECSA exit level outcome is listed along with a description of how it has been achieved and which chapters are relevant to it.

Exit level outcome	Description of outcome achievement	Relevant chapters
1. Problem solving	The problem of training and testing a machine learning model on small datasets, while still providing a sufficient level of personalisation to help a struggling first-time reader improve their reading, was identified. Previous solutions that achieved helpful results relied on much larger datasets, which are not always available or appropriate.	**Chapters 1.1, 1.2, 2.1.2, 3.3.1, 3.3.2, 3.3.3, 3.3.4, 4.3.1, 4.3.2, 4.3.3, 4.3.4, and 6.1**
2. Application of scientific and engineering knowledge	Engineering knowledge, including machine learning, mathematics and statistics, and computer science, was required to select and implement the machine learning models to compare in order to find the most appropriate for small datasets.	**Chapters 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.5, 4.3.1, 4.3.2, 4.3.3, and 4.3.4**

Table B.2: Each ECSA exit level outcome is listed along with a description of how it has been achieved and which chapters are relevant to it. (continued)

Exit level outcome	Description of outcome achievement	Relevant chapters
3. Engineering Design	Feature vectors were designed to effectively highlight similarities between sentences, machine learning models were selected and implemented to be suitable for training and testing on small datasets, and a sentence generation algorithm was designed to create personalised sentence recommendations for each agent.	**Chapter 1.2, 3.2, 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.4, 4.2, 4.3.1, 4.3.2, 4.3.3, 4.3.4, and 4.4**
4. Investigations, experiments and data analysis	The efficacy of different machine learning models in making predictions based on small datasets was investigated, and the accuracy, precision and recall of different models was evaluated. The sentences generated by the context-free grammars were analysed to gauge how relevant they were to each agent's individual errors.	**Chapter 4.3.1, 4.3.2, 4.3.3, 4.3.4, 5.1, 5.2, and 6.1**
5. Engineering methods, skills and tools, including Information Technology	The entire system is software-based, implemented in Python. Software libraries were used for the machine learning models, context-free grammars and sentence generation.	**Chapter 4.2, 4.3.1, 4.3.2, 4.3.3, 4.3.4, and 4.4**

Table B.3: Each ECSA exit level outcome is listed along with a description of how it has been achieved and which chapters are relevant to it. (continued)

Exit level outcome	Description of outcome achievement	Relevant chapters
6. Professional and technical communication	This report is written in the style of a professional, technical report, making use of LaTeX as a writing and document preparation tool.	**Chapter 1.1, 1.2, 1.3, 1.4, 2.1.1, 2.1.2, 2.2, 2.3, 3.1, 3.2, 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.4, 3.5, 3.6, 4.1, 4.2, 4.3.1, 4.3.2, 4.3.3, 4.3.4, 4.4, 4.5, 5.1, 5.2, 5.3, 6.1, and 6.2**
8. Individual work	All work done on this project was done individually, with Dr H. Kamper acting in a supervisory capacity.	**Chapter 1.1, 1.2, 1.3, 1.4, 2.1.1, 2.1.2, 2.2, 2.3, 3.1, 3.2, 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.4, 3.5, 3.6, 4.1, 4.2, 4.3.1, 4.3.2, 4.3.3, 4.3.4, 4.4, 4.5, 5.1, 5.2, 5.3, 6.1, and 6.2**
9. Independent Learning Ability	Various new concepts, such as analysis of children's reading miscues, bag-of-words feature vectors, constructing agents on which to train models, decision trees, the *k*-nearest neighbours algorithm, neural networks, context-free grammars and sentence generation were learned and implemented independently for this project. The non-traditional application of machine learning models to small datasets demonstrates independent thinking.	**Chapter 2.1.1, 2.1.2, 2.2, 3.2, 3.1, 3.3.1, 3.3.3, 3.3.4, 3.4, 4.2, 4.1, 4.3.1, 4.3.3, 4.3.4, 4.4, 5.1, 5.2, and 6.1**