VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Automated Essay Scoring Using Machine Learning with Manual Feature Engineering And Regression Models

James Ian Rowles

Supervisors: Sharon Gao

Submitted in partial fulfilment of the requirements for
Bachelor of Engineering with Honours.

Abstract

The primary aim of this project is to grade students' essays from The English Proficiency Program automatically. This research aims to address this issue by examining an approach using regression models and feature engineering.

Contents

Acknowledgements I want to thank my supervisors Sharon Gao and Peter Gu for their support and guidance throughout this project. I would also like to express my gratitude towards my supervisors for providing me with this opportunity. As an individual who highly values the pursuit of learning and has played both roles - teacher and student. I feel very privileged to be given the opportunity to contribute back to this community. Lastly, I would like to thank my examiners for their contribution to completing this project.

Appendix A

Introduction

A.1 Context

Hundreds of students take the English Proficiency Programme (EPP) every year. Its aim is to help students develop their English in a welcoming learning setting. In addition, the EPP can train a pupil for more English-language academic training. The EPP facilitates a test to assess a student's ability to see if they would meet the entry criteria for their specific undergraduate or postgraduate course. This test consists of multiple components, including a multi-choice and a writing component. Based on how they perform overall dictates whether or not they will enter their proposed course [1]. However, if their overall performance is unsatisfactory, they will stay with the EEP to build their skills. In this case, it is beneficial for the academic staff of the EPP to understand which areas the student is struggling in to provide more tailored support for that student.

A.2 Problem

The problem that exists here is that manually marking the writing component of this test takes too long and, as a result, is not included when making the final decision regarding whether or not a student has met their respective course's entry criteria. Excluding this component means that students are potentially not assessed as accurately as they could be. As a result,they can not receive the most optimal academic experience.

A.3 Purpose

The purpose of this project is to create a system that can automate this task. Roughly, this looks like a system that can use an essay as input and, in return, provide quantifiable metrics regarding the essay's use of the English written language.

A.4 Goals

The goals for the solution are for it to be relatively fast, precise enough, and easy to use and understand.

Speed As speed is a leading contributor to the problem, this plays a massive part in the goals of the solution. Specifically, this means that it must be faster than if a human marker were doing it.

Accuracy We need the system to predict similar marks to what a human rater might produce in terms of accuracy. However, as mentioned previously, the writing component is not used currently due to time constraints. As a result, even if the system were roughly accurate, it would still provide valuable insight to the academic staff. For example, its results could be used "with a grain of salt" and in combination with other assessment metrics at the final-judgment stage of the EPP program.

Easy of Use In terms of ease of use, for the final product, this may be defined as having a polished user experience held together with simple, maintainable network architecture. For this project's scope, I want to prioritize keeping the system explainable or understandable so that non-technical stakeholders have insight into the nature of the system - how it works, why it does things. Additionally, I want the system to be easily understood by a technical stakeholder such as the next student to work on this project.

A.5 Approach

Initially, we saw there being two approaches to the problem.

Approach 1 The first approach would provide a binary output to indicate whether the student had a good enough proficiency to enter their proposed university course or needed to continue with the EPP to improve their English writing proficiency further.

Approach 2 The second approach would aim to go a step further by providing a continuous output that would provide insight into the extent of a student's overall writing proficiency and their categorical proficiency in areas such as; grammar, vocab, or ideas. With this approach, not only do we receive enough information to assess if they can enter their proposed university course or not. We can also go a step further by having insight into the extent to which they may or may not do so. For example, if the student needs to get an overall score of 4 to progress and receive a 3, this might lead us to believe that maybe they are not ready to progress. Also, with this approach, we get the same continuous-type output for specific categories. This means that, for the students who do not progress, we have insight into the areas that a student may be struggling with and the extent to which they may be struggling with them. This, then, allows for more tailed academic support. The same goes for areas that a student might be proficient in.

The latter provides the most information and consequently the most benefit for all stakeholders. So right from the offset, this was the approach that I wanted to take, provided it was a feasible pursuit.

Appendix B

Background

To investigate the feasibility and potential pathways, I started by looking into general applied machine learning practices and academic papers about already existing systems like the one I was pursuing. Not to my surprise, I found that this process of automating essay scoring has been a subject of study for a while, specifically within education and business. An early note-able pursuit within this field dates back to the 1960s where Ellis Batten Page's PEG, Project Essay Scorer, began to gain public and scientific momentum and acceptance [2]. The systems that are products of this area are called AES's - Automatic Essay Scorers. A good AES can deliver scores that are relatively similar to human ratings. For example, the Educational Testing Service's e-rater system, also known as "e-rater," can produce accurate results over 90% percent of the time [3].

B.1 Machine Learning

AES systems have incorporated a sub-field of Artificial intelligence, Machine Learning, from early as 2013 [4]. Predictive Machine Learning is the science of making computers intelligent enough to make their own decisions. It has been around for a long time, but it has become powerful enough that it can now be used on various problems in recent years. Predictive Machine learning is generally used for tasks where it is beneficial for a computer to understand a relationship between an input and an output for a specific problem [5]. This is often used to use this relationship understanding to synthesize an output when only given an input. In our case, this relationship is between a text, and it is the corresponding score. We want to refine this relationship between the input and output to be given an unmarked essay and produce a score.

B.2 Models

This relationship is called a model and is defined as a description of a system defined with mathematical concepts and language [6]. The model of an AES is a critical component of the overall solution. There are several types within the field of Machine Learning and AES. They can be divided into two categories. The first category consists of relatively simple models that employ statistical calculations such as Regression or Classification. Regression can be used to predict continuous values, for example, a score of "5.0" or "3.1" [7]. Classification can be used to identify, separate, or categorise values into discrete values, such as "true" or "false" [8]. The second category consists of relatively more complex models. These models work by simulating a network of interconnected nodes that work similarly to neurons in the

human brain. To recognize and learn correlations within the data. They can then be used to predict continuous or discrete data [9].

B.3 Feature Extraction

Each model requires its input to be in a process-able format and descriptive of the problem before it can mathematically evaluate it. This process of optimizing the input to a more interpretable format is called Feature Extraction [10]]. When using a Neural Network model, this step can be skipped entirely [11]. However, when working with Regression and Classification models, this step must be performed manually using Manual Feature Engineering. This requires identifying and programmatically developing Features from an essay that are thought to indicate the skills being assessed [12. In our case, a feature can be thought of as a characteristic, property, or attribute of an essay. For example, the number of words in the text, the number of grammatical errors, or the average length of the words used. This process is done one feature at a time, requires domain knowledge, and can be time-consuming and error-prone.

B.4 Model Choice

Prediction Type The output alone influenced my decision to deviate from a classification model. As, fundamentally, classification is about predicting a categorical variable. However, this type of output variable could be used to address the first approach mentioned in the introduction by, for example, classifying a student into either a ”doing well” or ”not doing well” group for a marking-criteria category. Neural and Regression models can do better by producing continuous outputs or predictions that align better with this project's preferred approach as they would produce a continuous output for a marking-criteria category.

Nature of Relationship The nature of the relationship that is being investigated is also a crucial consideration. In some applications, neural network models can perform better than linear regression models when there are non-linearities involved. This is because neural networks, when appropriately configured, can understand highly complex and convoluted non-linear relationships. [13]. Non-linearity is a term used in statistics to describe a situation where there is no straight-line or direct relationship between two variables. In a non-linear relationship, changes in the output do not directly change to changes in any of the inputs [14]. In this study, I expect to observe linear relationships between the features and the output due to the nature of the problem of this project. I would expect to see that as the number of grammatical errors increases, the computer-generated score will decrease. This influenced my decision to lean toward a Regression model as they can perform well with linear relationships[x]. Additionally, they can do so with less configuration and, therefore, domain knowledge [x].

The Data The data available also plays a considerable role in narrowing down model options. As its size and relevance can influence a models' ability to synthesize the relationship in question. This is especially true for Neural Network models due to their better generalised ability to learn. As for where more data is available, better results can be observed from Neural network models [x]. However, in cases where this is not true, a Regression Model can be a better choice. For the duration of this project, nine datasets were used. Of

these datasets, only one directly represented the problem and was not available for the entire duration of the project either. For these reasons, I saw a Regression Model being a better choice.

Feature Extraction Feature Extraction is another critical consideration when choosing a model. When considering a Neural model, the possibility of skipping the process of Manual Feature Engineering entirely can sound like an immediate win. However, despite Manual Feature Engineering introducing another component into the mix, it can provide much value. Especially when dealing with smaller datasets. Lilja observed a performance decrease in their study with Neural Models. And suggested that human insight when working with linguistic features could be crucial when dealing with small sample sizes [15]. Additionally, I'm optimistic about Manual Feature Engineering as I believe most features will not require extensive domain knowledge and are also relatively subjective. For these reasons, from the perspective of the available data, I saw value in Manual Feature Engineering.

Explain-ability Lastly, explain-ability is an essential factor in this consideration as it aligns with the high-level goal of this project. Explainability is the concept that a machine learning model and its output can be explained in a way that "makes sense" to a human being at an acceptable level. When considering Manual Feature Engineering alone, there are already gains in explainability as the Feature Extraction step would be exposed and become the researchers' responsibility. This would mean that this step can be more easily explained to stakeholders. In addition, to provide insight, such as; what the features are and for which calculations they are being used for. Additionally, even without Manual Feature Engineering, Regression models are more accessible to explain than Neural Network models [x]. For these reasons, from the explaibabity, I saw a Regression with Manual Feature Engineering Model being a better choice.

As discussed in the previous sections, Manual Feature Engineering paired with a Regression Model is the most optimal starting point for this project.

Appendix C

Design

This project's scope is concerned with the development and experimentation of the two core components that make up this system. These are The Feature Extractor and The Predictor. Although this project is relatively immature, I have paid some respects to the theoretical end-solution at each project stage. This was important as it relates to the motivation behind the project - to provide the stakeholders with a usable tool. In saying this, the following diagram is a naive representation of the theoretical end-solution or product.

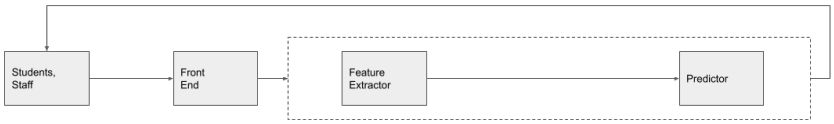

Figure C.1: AES as a Product

As for the scope of this project, as mentioned previously, it is concerned with the development and experimentation of the Feature Extractor and The Predictor. Additionally, as with any machine learning endeavor, it heavily relies on relevant data to develop, test, and evaluate. Therefore, another critical in this investigation is The Prediction Configuration Generator (PCG). The PCG, paired with The Predictor and The Error Calculator, allowed for extensive exploration, insight, and evaluation of the system.

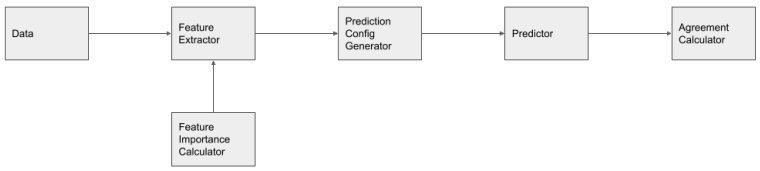

Figure C.2: AES in Development

C.1 The Data

Preparing the data for the Feature Extractor can be broken down into three main stages - discovery, digitising, and processing. The product of these stages is the structured digital representation of an essay given to the Feature Extractor.

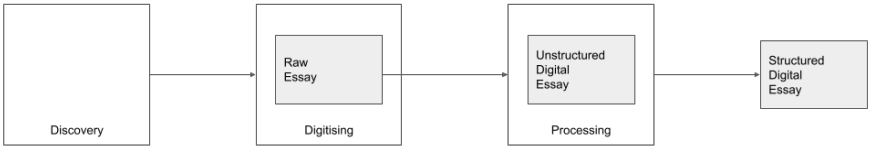

Figure C.3: The Data: Stages

C.1.1 Discovery

The discovery step was mainly concerned with discovering relevant data sets to be used for the project. The Hewlett foundation competition data was used in the early stages of the project. As it progressed, the Victoria University of Wellington: English Proficiency Program data was acquired.

Dataset	Hewlett								VUW
Total Essays	1,785	1,800	1,726	1,772	1,805	1,800	1,730	918	113
Average Words	350	350	150	150	150	150	250	650	376

Table C.1: Dataset Statistical Overview

The Victoria University of Wellington: English Proficiency Program The Victoria University of Wellington: English Proficiency Program (VUW EPP) dataset consists of 113 essays with scores for each of the following categories, which were triple marked: grammar, vocab, flow, ideas, coherence, overall. This dataset initially consisted of two separate sets and was joined into one. Each set is from a trimester of the written assessment component of the EPP program.

The Hewlett Foundation The Hewlett Foundation dataset Competition consists of eight essay sets. Each of the essay collections was born from a particular prompt. The length of selected essays varies from 150 to 550 words per response. Some of the writings rely on information from sources, while others do not. All of the replies were submitted by kids in grades 7 through 10, and they ranged in age from 7 to 10. All essays were double-scored and assessed by hand. Thus, each of the eight data sets is distinct in its way.

C.1.2 Digitisation

The digitisation stage mainly consisted of converting handwritten essays to digital word documents. The essay format from the data from the Victoria University of Wellington: English Proficiency Program was handwritten. This was done by manually transcribing each essay using text-to-speech to multiple word documents. The scores from this dataset were in the format of a printed document. This was digitized manually through data entry to a spreadsheet document.

C.1.3 Processing

The processing stage consisted of converting unstructured-digital data into structured data. Throughout this task, I tried to automate as much as possible to save time and reduce the

risk of human error. The implementation required for this stage is discussed further in the implementation section.

The data in this project consists of two common format types - Comma-Separated Values (CSV) and Tab-Separated Values (TSV). The main difference between them is simply the identifier that the information is separated with - either a comma or a tab. They are both widely used for many purposes and are primarily encountered in spreadsheets and databases. I chose to use them because they are widely used and, as a result, can be easily interpreted by many different programs or tools, such as spreadsheet processors. These were used to manipulate the data manually—also, Python, where libraries already existed to easily interpret and parse data with this structure.

The target structure is to have one essay per row, with its respective information following on the same row, in the following order: the id, the text of the essay, and then the scores associated with that essay. Of course, the scores will vary depending on the dataset.

id	essay	scores				

Table C.2: Structured Data Table Format

C.2 The Feature Extractor

The Feature Extractor is responsible for identifying, calculating, categorizing, and numerically representing the ”features” of a given essay so that they can be inputted into other components to produce computer-generated scores. For example, the following figure depicts how The Feature Extractor takes a structured essay as input and produces multiple feature categories.

A feature category is simply a TSV file containing a row for every essay from the inputted structured essay data and columns for the respective ids, scores, and features relevant to the category. For example, for most executions, the feature extractor will output six TSV files for each category. Each file will contain a row for every essay, although the columns will vary as they will only contain the features for their respective category. The figure below is a visual representation of this table structure.

id	features								scores			

Table C.3: Feature Category Table Format

C.2.1 Categories

The categories are based on the EEP marking criteria to optimize the prediction of these same categories by organizing each feature into a group or category based on their relevance with these same categories. For example, categorizing the count of grammatical errors of an essay into the grammar feature category will very likely aid in depicting an essay’s proficiency within this category and, therefore, the system’s ability to predict such a mark.

C.2.2 Resources

The Feature Extractor uses three predefined resources to help calculate several specific functions. These resources are described below.

Headwords and Basewords The Headwords and Basewords resource consist of 25,000 headwords and basewords from The Corpus of Contemporary American English (COCA) and the British National Corpus (BNC). Headwords are essential to the core meaning of a phrase, for example, ”create.” Basewords can be thought of as variations of this word, for example, ”create,” ”creation,” ”creative.” The COCA and the BNC complement each other nicely, and they are only large, well-balanced corpora of English that are publicly available. The BNC has better coverage of informal, everyday conversation, while COCA is much larger and more recent, which has important implications for the quantity and quality of the data overall. The lists are designed primarily for learners of English as a foreign language [16].

Stopwords The Stopwords resource consists of a list of stopwords from the Natural Language Toolkit Python library. Stopwords are a set of commonly used words in a language. Examples of stop words in English are ”a”, ”the”, ”is”, ”are”, and ”etc”.

POS Trigrams The POS Trigrams resource consists of a list of Trigrams. A trigram is a contiguous sequence of three items from a given sample of text or speech. These items can be phonemes, syllables, letters, words, or base pairs according to [based] on the application. The items in this list only consist of part-of-speech identifiers, such as; nouns, verbs, adjectives, adverbs. This list was imported from the Travis Moore study. He found that from a particular set of POS trigrams, there was a correlation between the amount of these trigrams an essay contained and its score [17].

C.2.3 The Features

For every essay, the following features are calculated and categorized. Below is a summary of every feature and its respective category. More information about the implementation of the extraction or calculation of these can be found in the feature extraction section of the implementation section. Most features from this section are derived from the Travis Moore study [17].

Miscellaneous

This category contains some elementary features that I chose not to categorize, but instead, they are added to every category. This category is not part of the exported feature categories. The produced features include:

- The total words
- The total sentences
- The total words per sentence
- The sentences per 100 words

Grammar

Transition Words Transition words are words that help connect or link ideas, phrases, sentences, or paragraphs. These words help the reader smoothly through ideas by creating a bridge between them. The transition words of the essay are used to produce the following features:

- The percentage of the transition words in the essay out of the Transitions Set.

Parts of Speech Trigrams The POS Trigrams Resource is used to find matches in the essay, and then the count of matches is the percentage of Parts of Speech trigrams found in the essay from the ”relevant trigrams set.”

Vocab

Difficult Words The difficult words are discovered using the Python Textstat library (explained further below) and are used to produce the following features:

- The total difficult words in the essay

Headwords and Basewords The Headwords and Basewords resource is separated into multiple lists. Then, for each list, the words from the list and the essay are compared to produce the following features:

- The words in both divided by the list length
- The words in both divided by the total words of the essay

Unique Words Unique words are defined as being the only one of their kind and are used to produce the following features:

- The total unique words in the essay
- The average unique words (total unique divided by the total words)
- The number of unique words per 100 words

Flow

Readability Counts Textstat is used to produce the following counts:

- Total sentences
- Total characters
- Total letters

Syllables The syllables in each word of the essay are used to process the following features:

- The total syllables
- The number of words with a syllable count greater than two
- The number of words with a syllable count equal to one

Reading Time The time it would take to read the essay

Readability Consensus The estimated school grade level required to understand the text

Readability Scores Using Textstat's readability measures, the following features are produced:

- The Flesch Reading Ease
- The Flesch-Kincaid Grade Level
- The Fog Scale Returns the FOG
- Spache Readability Formula
- The SMOG Index
- The Coleman-Liau Index
- Automated Readability Index
- Linear Write Formula
- Dale-Chall Readability Score

Coherence

Lemmas A lemma is a canonical form, dictionary form, or citation form of a set of words. The lemmas of the essay are used to produce the following features:

- Total Lemmas
- Total Lemmas from Unique Words
- Total Brigrams from Lemmas
- Total Brigrams from Lemmas from Unique Words
- Total Trigrams from Lemmas
- Total Trigrams from Lemmas from Unique Words

Content Words Content words, in linguistics, are words that possess semantic content and contribute to the meaning of the sentence in which they occur. The content words are discovered by removing all stopwords from the text. The content words of the essay are used to produce the following features:

- The total content words
- The total unique content words
- The percentage of unique content words out of the non-unique content words found

Function Words Function words are words that have little lexical meaning or have ambiguous meaning and express grammatical relationships among other words within a sentence or specify the attitude or mood of the speaker. They signal the structural relationships that words have to one another and are the glue that holds sentences together. Thus they form essential elements in the structures of sentences. The function words of the essay are used to produce the following features:

- The total function words
- The total unique function words
- The percentage of unique function words out of the non-unique functions found

Nouns

- The nouns of the essay are used to produce the following features:
- The total nouns
- The total unique nouns
- The percentage of unique nouns out of the non-unique nouns found

Determiners A determiner is a modifying word that determines the kind of reference a noun or noun group has, for example, ”a,” ”the,” ”every.” The determiners of the essay are used to produce the following features:

- The total determiner
- The total unique determiner
- The percentage of unique determiner out of the non-unique determiners found

Conjunctions A conjunction is a word used to connect clauses or sentences or coordinate words in the same clause. For example, ”but”, ”if”. The conjunctions of the essay are used to produce the following features:

- The total conjunctions
- The total unique conjunctions
- The percentage of unique conjunctions out of the non-unique conjunctions found

Pronouns A pronoun is a word that can function as a noun phrase used by itself, and that refers to the participants in the discourse, for example, ”I,” ”you,” or to someone or something mentioned elsewhere in the discourse, for example, ”she,” ”it,” ”this.” The pronouns of the essay are used to produce the following features:

- The total pronouns
- The total unique pronouns
- The percentage of unique pronouns out of the non-unique pronouns found

Ideas

This category does not contain any unique features.

Overall

This category contains the features from all of the previous categories.

C.3 The Predictor

The Predictor is responsible for producing computer-generated scores. As described in the following figure, it takes a prediction configuration as input and produces scores based on this configuration.

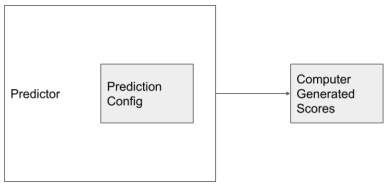

Figure C.4: The Predictor

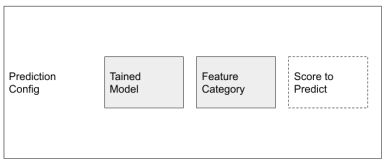

Figure C.5: Prediction Configuration

C.3.1 Prediction Configuration

A Prediction Configuration, as seen below, is composed of a trained model, a feature category, and the target score to predict from the feature category. A trained model is a model which has an enhanced understanding of a relationship between an input and an output for a specific problem by being given examples of this problem from a specific dataset.

C.3.2 The Prediction Configuration Generator

The Prediction Configuration Generator (PCG) is responsible for producing multiple prediction configuration objects. It can also be used to produce a single prediction configuration object, which may be useful after the evaluation stage of this project.

For the experimental stage of this project, it was advantageous to produce prediction configuration permutations because it allowed us to explore the a variety of Prediction Configurations.

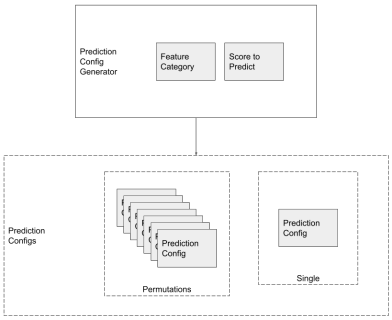

Figure C.6: PCG: Permutator

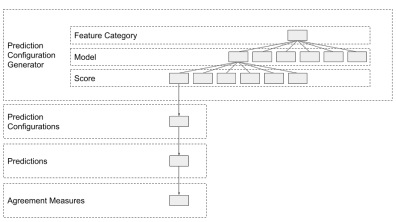

Figure C.7: PCG: Pipeline

Appendix D

Implementation

D.1 Development

Initial programs were made using google-colab. This platform was beneficial in the exploration stage for programmatically investigating Natural Language Processing and Machine Learning. Following this, all implementation was done with Python in Visual Studio Code. Several libraries have been incorporated, such as Natural Language Toolkit, Language Check, Text Stat, Pandas, Matlab, Sklearn. Executions were orchestrated using Bash. Amazon's Elastic Cloud Compute (EC2) service was also used for executions. Amazon Elastic Compute Cloud is a part of Amazon.com's cloud-computing platform, Amazon Web Services, which allows users to rent virtual computers to run their computer applications.

D.1.1 Structure

The project's root is the "AES" folder, the requirements.txt, and the "runner.sh" which is the batch job orchestrator bash script. Also, all execution log files will appear at this level of the project. My reason for only having these files at this level is that the project is more runnable than a codebase. So all the things you need to execute any component can be done from here, and all log files will appear here without the clutter or mechanics of the python files that do the automatic scoring getting in the way.

Inside the AES folder is where all of the python code files, data, and artifacts live. The folders at this level are split into categories based on the artifacts that the project produces or consumes, such as; agreement tests, feature categories, headwords, and baseboards. Also, at this level are all of the python files. Again, this is pure to accommodate working with multiple files in Python. Splitting up the code as much as possible was a priority of mine as I wanted to keep the code abseils as manual as possible so that each opponent was easy to understand and there develop and debug. Although this does mean that there are several main files all at one place in eh director, this was worth it as I wanted to avoid creating a monolith. Again this was for my benefit as well as the future students of this project. Additionally, having well-defined separated responsibilities allows for a more straightforward explanation of the system to non-technical stakeholders. The location or folder structure of most artifacts, can be seen in the figure. I made this early design choice to standardise the general structure for any given dataset, of any version, being run against any component. This was a benefit as there is a lot of cross-pollination of the component ts. As they all produce and consume different artifices for each other. So having some unified rules around this was helpful. This is primarily due to the nature of my modularised system, where each component is frequently importing and exporting things.

D.1.2 Executions

An execution can be thought of as running a component against a dataset. For example, common execution goals include running The Feature Extractor against the VUW EPP data to produce Feature Categories or running The PCG, The Predictor, and The Agreement Calculator against the Hewlett data to produce Agreement Measures. Often multiple executions are grouped and commenced with a single command, just like how a batch job might be done. A batch job can be considered a predefined group of processing actions with little or no interaction between the user and the system. In our case, this was often to run multiple datasets through the feature extractor, predictor, and agreement calculator.

D.1.3 Workflow

The workflow for conducting an execution is described in the following diagram.

Push This step is required to provide the EC2 instance with the relevant resources that it may require for a proposed execution. These resources are committed and pushed from the local machine and then are retrieved or pulled from git before an execution. For example, if the proposed execution is to generate features for a new dataset, this would require committing and pushing the respective dataset's TSV file, provided it is in the correct location.

SSH Once connected to the cloud instance. You can first pull from the remote to receive the newest version of the project on the instance. Following this, you can execute the runner script, which will execute predefined executions. After the bash script has been completed, you can then commit the newly created artifacts and push them to the remote to update them with the newly produced artifacts. While the bash script or runner is executing, you can observe the status of each job by viewing the associated log file.

This was made possible by incorporating extensive logging for all types of execution. The motivation and gains of this are discussed in the following sections.

As many components interact, it was beneficial to have some form of sanity checking to help mitigate elementary mistakes or impossibilities that might be unclear when programming or are based on invalid assumptions. In addition, having this level of visibility helped to ensure the system was operating as I expected. This was especially helpful when ensuring intended permutations were being made, which datasets were being used, as developing and validating this process could get quite overwhelming otherwise.

Having these extensive logs also provided more information when troubleshooting or investigating bugs by providing more information. As well as this, for most systems of its maturity, the time to implement some form of incremental run-state backup system is not worth it. In most cases, for systems of this maturity, if the system is running a job and it encounters an error, the system will crash entirely, and its current task's progress will be lost. And unfortunately, until that problem is resolved, the same task will continue to fail until it is directly attended to.

Additionally, I chose to export this log data to a file as, over time, the amount of information I was outputting exceeded the console's memory limit. As well as this, it was tough to retrieve log statements when running in the cloud after exiting an ssh session for a process executed. However, it also meant that I could quickly and conveniently review a job's status post-run.

Pull Once all executions have been completed or when you reconnect to the server, you can commit and push all new artifacts to the remote. Then you can exit the SSH session, and

from your local machine, pull from the remote to receive all the newly produced artifacts.

D.2 The Feature Extractor

The Feature extractor can be broken down into the following main stages.

1. Initialise In the Initialisation stage, one of the first things the AES will do is load any pre-existing resources that it may require, such as the headwords list, stop words list, and POS trigrams. Also, in this step, the system will initialize any necessary data structures, such as the lists for each feature category.

2. Import In the import stage, the system will read and translate the relevant essay TSV into a two-dimensional array. The location of the relevant structured essay, as mentioned previously, will be derived from the run parameters.

3. Then, for every essay, it will:

3.1. Process Identifiers Here, the feature extractor will process the essays identifiers to all feature categories.

3.2. Process The Features Processing a feature is done by performing a predefined calculation and then inputting the result of that calculation into the ”processing” method along with a category. This method will then add the value or result of the feature to a list for the specific category and the overall category. This was done simply as a naive approach to categorising. Going forward, this should be updated.

4. Export Feature Categories Lastly, the Feature Extractor will translate the categorised data structures into separate TSV fikes. In order to produce feature categories containing all essays with only the feature values specific to themselves.

D.2.1 Feature Importance Tests

The future tests work by comparing the overall future category, which contains all features, to every categorical score using many helper methods from sklearn similar to the prediction component.

D.3 The Predictor

From a programmatic perspective, the prediction component is relatively simple as it can be done in one line of code using the sklearn library. Therefore, in the final solution, this component will use the most optimal pre-trained model, for a category, from an imported ”.pkl” file. However, for the duration of this project, these models only needed to persist at run time for a single execution. Therefore they did not need to be exported.

D.3.1 The Prediction Configuration Generator

The Prediction Configuration Generator has two main methods. Permeate, and Run. They use the Prediction Configuration class, which stores a model, feature category, score to predict, and other fields. Having this simple class is super helpful for retaining an intended test's information.

The Permutate method mainly consists of a triple nested loop that loops through predefined models, feature categories, and scores to predict. The list of models remains the same for all execution types or configurations—however, the feature categories and scores to predict can vary depending on the execution params. For example, the scores to predict list for an "overall" run type only consist of the overall score. Each inner iteration will train a model, produce predictions, and compare them to the human-produce scores to produce agreement measures. One thing to note in this method is the outlier function.

The motivation behind this was to avoid processing any tremendous prediction values generated by the computer. Not only did this take a very long time, but it also meant that Prediction Configuration was very, wildly inaccurate anyway. So it did not need to be consumed by the agreement calculator and component to the current best QWK because it would not be a suitable candidate for having the best QWK.

The way these unusually high values are moderated is simply by calculating the mean of the prediction values list and comparing this to the mean of the actual human-generated scores list. If the mean is too big, the triple nested loop will end that iteration and start the logical next iteration.

To validate this, to make sure that I was not removing valuable test results. I implemented an "excluded "count and divided this by the size of the precision values list to get the percentage of Prediction Configurations being ignored. I have just left this debugging code in the system, and for all run's observed, the ignored configurations are a relatively small proportion.

D.3.2 The Agreement Calculator

The Agreement Calculator calculates the following measures: Quadratic Weighted Kappa, Pearson correlation coefficient, Adjacent Agreement Percentage, Exact Agreement Percentage. The logic and maths of the first two are mainly outsourced to sklean. The following two are both created manually. Most of these metrics do some mathematical operations on the human given scores and the computer-generated scores.

In the case of the adjacent-agreement measure, this is calculated by comparing these two values, calculating the absolute difference between them, checking if the difference is one (or less), and if so, incrementing the adjacent count variable by 1. This is done until all pairs of values are checked. Then, following this, a percentage is calculated by dividing the adjacent counts by the total pairs. In the case of the exact agreements, the producer is very similar. However, we check whether both values are the same instead of calculating a difference between the computer and human-generated scores. If they are, a counter is incremented, and a percentage out of all predictions is calculated from this, like the other measure.

Appendix E

Evaluation

E.1 The Feature Extractor

Within The Feature Extractor, simply identifying a feature is a major breakthrough. However, this is a time consuming and complex process. Beyond this, the next most valuable task that can be is understanding if a feature is in the correct Feature Category. However, due to the nature of feature engineering, as it solely relies on domain knowledge, there is likely always room for further optimisation or refinement, beyond the initial discovery or identification of them. As a result, it is worthwhile to investigate to what extent a feature is actually aiding the predictive model to verify our assumptions of the domain and potentially optimise our system.

Feature Importance Tests Feature Importance Tests can be useful for this. Feature Importance Tests refer to techniques that assign a score to input features based on how useful they are at predicting a target variable [x]. These give us an indication into whether or not our features are helping the predictor to synthesize a relationship between themselves and the score.

The results can then be used to delete the features with the lowest scores or keep the features with the highest scores. This can simplify the problem that is being modeled, speed up the modeling process, and potentially, improve the performance of the model [x].

F-Value The values or scores of each feature are generated through an F-Test. The F-Test is a form of hypothesis testing. Where a model is created with just a constant and another model is created with a constant and a feature. Then the least squared errors in both the models are compared to checks if the difference in errors between each model is significant or introduced by chance [x].

Below are the features for each category that have the highest importance. - I will compare the actual highest to the features that are currently being used. -

Vocab

Here we can see that 13 of the top 38 features are currently being used to predicting the Vocab score. This is 34% of this range. However, 8 of the top 19 are being used to predict the Vocab score. Which is an improvement, at 50%.

This is one of the higher proportions of strong features being used. However, the vocab category has the most amount of features, excluding the overall category, due to it consisting

of the headwords feature type. As the way this group of features is calculated is by producing a feature for each band of the headwords. Because of this, this would lead me to believe why there are a lot scores in the first two thirds of the tables.

Feature	Score	Feature	Score	Feature	Score
proportion_vocab_list[abandon]_headwords	**22.05833729**	nfunction_tokens	5.704050657	**proportion_vocab_list[abomasum]_total_words**	**3.095114627**
proportion_vocab_list[accent]_headwords	**16.40796885**	ncontent_types	5.219737871	**proportion_vocab_list[abomasum]_headwords**	**3.095114627**
polysyllabcount	15.71130959	n_trigram_lemma_types	5.118016095	**total_sentences**	**3.078611375**
difficult_words	**12.18352548**	**num_types**	**5.118016095**	ncontent_tokens	3.07783032
determiners_total	11.54240886	nlemma_types	5.118016095	ts_avg_len_sent	3.002896576
smog_index	9.174520901	n_bigram_lemma_types	5.118016095	automated_readability_index	2.779641652
determiners_per_words	8.446785067	crawford	4.81807535	monosyllabcount	2.616700091
text_standard	7.983828856	lexicon_count	4.468261165	conjunctions_total	2.568940715
syllable_count	7.092957374	**proportion_vocab_list[aardvark]_total_words**	**4.44829349**	nfunction_types	2.476570719
proportion_vocab_list[abaft]_headwords	**6.930334973**	**proportion_vocab_list[aardvark]_headwords**	**4.44829349**	pct_transitions	2.444242565
proportion_vocab_list[abaft]_total_words	**6.895066765**	nlemmas	4.300215352	spache_readability	2.440962422
proportion_vocab_list[abandon]_total_words	**6.703801959**	n_trigram_lemmas	4.300215352	**words_div_sentences**	**2.37079476**
proportion_vocab_list[absentminded]_total_words	**6.424550614**	n_bigram_lemmas	4.300215352	flesch_kincaid_grade	2.231981694
proportion_vocab_list[accent]_total_words	**6.358088016**	**proportion_vocab_list[a]_total_words**	**4.274963711**	**proportion_vocab_list[abashed]_total_words**	**2.144350221**
function_ttr	6.283303783	**total_words**	**4.246082192**	**proportion_vocab_list[abashed]_headwords**	**2.144350221**
grammar_errors_ltp_by_words	5.873555877	pct_rel_trigrams	4.07939033	coleman_liau_index	1.978447928
letter_count	5.856224883	pronouns_total	3.865049524	pronouns_noun_ratio	1.765296516
reading_time	5.742222651	ts_avg_syllab_per_word	3.349429602	pronouns_density	1.74673426
char_count	5.742099891	sentence_count	3.194236862	**proportion_vocab_list[a]_headwords**	**1.739721638**

Table E.1: Vocab Feature Importance Tests

Ideas

Here we can see that 2 of the top 38 features are currently being used to predict the Vocab score. This is 5% of this range, which is very low. Additionally, 1 of the top 19 are being used to predict the Vocab score. Which is also 5%.

The vocab category has the most amount of features, excluding the overall category, due to it consisting of the headwords feature type. As the way this group of features is calculated is by producing a feature for each band of the headwords. Because of this, this would lead me to believe why there are a lot scores in the first two thirds of the tables.

These results are potentially some of the most valuable. As this category was and the most neglected due to its complexity. The marking criteria definition for the high end scores of the Ideas category is "Development of ideas is deep and convincing." For this reason, the category only consisted of the "miscellaneous" features which were added to all feature categories. These include: total words total sentences and sentence density.

Feature	Score	Feature	Score	Feature	Score
determiners_total	23.98975321	monosyllabcount	17.06445867	pronouns_density	2.86945213
syllable_count	22.65767743	polysyllabcount	16.01149404	content_ttr	2.796727653
letter_count	21.76693064	proportion_vocab_list[a]_headwords	15.05858081	ttr_nouns	2.708399971
reading_time	21.50866321	nouns_total	14.93762526	conjunctions_unique	2.399994321
char_count	21.50722445	ncontent_types	14.48729494	pronouns_unique	2.399994321
nfunction_tokens	21.03021095	pct_transitions	14.19203036	ts_avg_sent_per_word	2.38213524
nfunction_types	20.2273507	proportion_vocab_list[a]_total_words	12.68306971	proportion_vocab_list[abomasum]_total_words	2.296469714
function_ttr	20.22092973	pronouns_total	11.80391532	proportion_vocab_list[abomasum]_headwords	2.296469714
lexicon_count	20.04425343	difficult_words	11.43922304	proportion_vocab_list[aal]_total_words	1.736032636
nlemmas	19.87833192	conjunctions_total	9.925480901	**sent_density**	**1.494882434**
n_bigram_lemmas	19.87833192	proportion_vocab_list[accent]_headwords	9.86627013	proportion_vocab_list[a1]_total_words	1.231983429
n_trigram_lemmas	19.87833192	**total_sentences**	**9.683989634**	pronouns_noun_ratio	1.22992392
total_words	**19.58377946**	sentence_count	9.665452001	text_standard	1.229178768
pct_rel_trigrams	18.84225246	ttrr	9.10015428	linsear_write_formula	1.195434814
ncontent_tokens	17.70236467	unique_words	9.080102356	proportion_vocab_list[absentminded]_headwords	1.191540578
n_trigram_lemma_types	17.21977037	nouns_unique	7.210491094	proportion_vocab_list[abalone]_total_words	1.10443711
num_types	17.21977037	proportion_vocab_list[abandon]_headwords	4.724206672	proportion_vocab_list[abalone]_headwords	1.10443711
nlemma_types	17.21977037	determiners_per_words	4.698053181	dale_chall_readability_score	1.059783983
n_bigram_lemma_types	17.21977037	proportion_vocab_list[aal]_headwords	4.37518797	proportion_vocab_list[abandon]_total_words	0.7944183939

Table E.2: Ideas Feature Importance Tests

Grammar

Here we can see that 5 of the top 38 features are currently being used to predicting the Grammar score. This is 13% of the most important features for this range. For the next

range, we see similar results as 3 of the top 19 are being used to predict the Grammar score. Which is a slight at 15%.

This feature only consisted of three features although I was very confident in the grammar errors feature as I assumed this would be very strongly related. This assumption has (lucky) proven correct in the results as it is the strongest feature. Additionally, the remaining grammar-unique features from this category are within the top 19 features.

However, despite the fact that the current features for this category are important. There simply is just enough of them.

Feature	Score	Feature	Score	Feature	Score
grammar_errors_ltp_by_words	**13.86887276**	n_trigram_lemmas	8.428847883	proportion_vocab_list[abomasum]_total_words	3.447080979
polysyllabcount	13.5453572	proportion_vocab_list[accent]_headwords	8.335462779	proportion_vocab_list[abomasum]_headwords	3.447080979
determiners_total	11.99318654	**total_words**	**8.332141187**	smog_index	2.652814039
syllable_count	11.06434199	ncontent_types	7.590394235	proportion_vocab_list[abnormal]_total_words	2.578601176
letter_count	9.777308962	ncontent_tokens	7.000690642	proportion_vocab_list[aback]_headwords	2.451993818
nfunction_tokens	9.676747576	proportion_vocab_list[a]_headwords	6.970460707	proportion_vocab_list[aback]_total_words	2.412191877
reading_time	9.589911019	monosyllabcount	6.821351862	proportion_vocab_list[abaft]_total_words	2.159653449
char_count	9.587821097	proportion_vocab_list[abnormal]_headwords	6.794462298	proportion_vocab_list[abaft]_headwords	2.037732545
difficult_words	9.085364454	function_ttr	5.868971179	nouns_unique	2.03145763
pct_rel_trigrams	**8.874333893**	proportion_vocab_list[abandon]_headwords	5.325501914	proportion_vocab_list[absentminded]_total_words	2.008426737
nfunction_types	8.78489224	**total_sentences**	**4.777733853**	proportion_vocab_list[abashed]_total_words	1.807824726
pct_transitions	**8.683182431**	sentence_count	4.665500367	proportion_vocab_list[abashed]_headwords	1.807824726
lexicon_count	8.642680781	pronouns_total	3.991921505	text_standard	1.679480615
n_trigram_lemma_types	8.633375961	determiners_per_words	3.877961209	ttrr	1.666647575
num_types	8.633375961	conjunctions_total	3.595950896	crawford	1.66445437
nlemma_types	8.633375961	proportion_vocab_list[a]_total_words	3.561214109	unique_words	1.496118802
n_bigram_lemma_types	8.633375961	nouns_total	3.448940441	linsear_write_formula	1.48898339
nlemmas	8.428847883	proportion_vocab_list[abet]_total_words	3.447080979	proportion_vocab_list[a1]_total_words	1.389620477
n_bigram_lemmas	8.428847883	proportion_vocab_list[abet]_headwords	3.447080979	ts_avg_sent_per_word	0.7677552826

Table E.3: Grammar Feature Importance Tests

Flow

Here we can see that 10 of the top 38 features are currently being used to predict the Flow score. This is 26% of the most important features for this range. For the next range, we see the exact same results as 5 of the top 19 are being used to predict the Grammar score. Which is 26% of this range.

This feature consists of a huge amount of imported libraries that calculate readability measures for the text. For this reason, I find it quite surprising that there is not a greater percentage for these measures with these ranges. Five of the twelve readability measures are evident in the top 38 features.

Feature	Score	Feature	Score	Feature	Score
grammar_errors_ltp_by_words	10.52322172	**reading_time**	**3.446741393**	**linsear_write_formula**	**2.179626868**
proportion_vocab_list[accent]_headwords	8.616507797	**char_count**	**3.445241765**	proportion_vocab_list[a]_total_words	1.970600376
proportion_vocab_list[abaft]_headwords	7.174452719	nfunction_types	3.25497433	proportion_vocab_list[aback]_total_words	1.786283529
proportion_vocab_list[abaft]_total_words	7.137848797	**crawford**	**3.153343606**	**monosyllabcount**	**1.743636837**
pct_transitions	5.952052458	proportion_vocab_list[abet]_total_words	2.785493827	proportion_vocab_list[aal]_total_words	1.605845889
polysyllabcount	**5.908466276**	proportion_vocab_list[abet]_headwords	2.785493827	proportion_vocab_list[abrasion]_headwords	1.539322698
difficult_words	5.813492969	**lexicon_count**	**2.775661365**	pronouns_total	1.371066736
text_standard	**4.888289364**	proportion_vocab_list[a]_headwords	2.771527007	**coleman_liau_index**	**1.332429938**
proportion_vocab_list[abandon]_headwords	4.575020547	**total_words**	**2.733166972**	proportion_vocab_list[aback]_headwords	1.320345559
n_trigram_lemma_types	4.263709123	nlemmas	2.66E+00	**total_sentences**	**1.119213959**
num_types	4.263709123	n_bigram_lemmas	2.661185087	**sentence_count**	**1.118883069**
nlemma_types	4.263709123	n_trigram_lemmas	2.661185087	proportion_vocab_list[aardvark]_total_words	0.9790015297
n_bigram_lemma_types	4.263709123	ncontent_tokens	2.62501352	proportion_vocab_list[aardvark]_headwords	0.9790015297
syllable_count	**4.156071732**	determiners_total	2.558186665	sent_density	0.9545624739
pct_rel_trigrams	4.011612142	nfunction_tokens	2.530773085	proportion_vocab_list[abate]_total_words	0.9144704138
ncontent_types	4.011139066	proportion_vocab_list[absentminded]_total_words	2.46882231	proportion_vocab_list[abnormal]_headwords	0.90326088
function_ttr	3.796428186	proportion_vocab_list[accent]_total_words	2.23868793	proportion_vocab_list[abate]_headwords	0.8364237459
letter_count	**3.528072315**	proportion_vocab_list[abashed]_total_words	2.215948697	nouns_total	0.8331892199
smog_index	**3.499672392**	proportion_vocab_list[abashed]_headwords	2.215948697	conjunctions_unique	0.8321611094

Table E.4: Flow Feature Importance Tests

Coherence

Here we can see that 16 of the top 38 features are currently being used to predict the Coherence score. This is 42% of this range, which is the highest percentage of all of categories

for this range (excluding the overall category). Additionally, 8 of the top 19 are being used to predict the Coherence score. Which is the again, the highest percentage of all of the categories for this range (excluding the overall category), at 50%.

This score has the features with the highest percentage of important features in both bands. However, it does have the second largest amount of features, excluding the overall category. As the way this group of features is calculated is by producing a feature for each band of the headwords. Because of this, this would lead me to believe why there are a lot scores in the first two thirds of the tables.

Feature	Score
pct_transitions	20.68884211
polysyllabcount	17.13412183
pct_rel_trigrams	14.92687473
syllable_count	13.17692848
nfunction_tokens	**11.94914458**
letter_count	11.73462468
proportion_vocab_list[abandon]_headwords	11.47026469
reading_time	11.40182309
char_count	11.39842839
function_ttr	**10.84975839**
grammar_errors_ltp_by_words	10.77253679
lexicon_count	10.08744734
determiners_total	**9.994962745**
nlemmas	**9.723691954**
n_trigram_lemmas	**9.723691954**
n_bigram_lemmas	**9.723691954**
total_words	**9.591339672**
proportion_vocab_list[aback]_total_words	8.807317784
nfunction_types	**8.672600025**

Feature	Score
pronouns_total	**8.5764678**
monosyllabcount	8.14921556
n_trigram_lemma_types	**7.57892787**
num_types	7.57892787
nlemma_types	**7.57892787**
n_bigram_lemma_types	**7.57892787**
ncontent_tokens	**7.555855104**
proportion_vocab_list[a]_total_words	7.412367857
difficult_words	6.882954381
proportion_vocab_list[aback]_headwords	6.814253694
proportion_vocab_list[a]_headwords	6.698095193
ncontent_types	**6.48683046**
text_standard	5.44173295
proportion_vocab_list[accent]_headwords	4.88851034
smog_index	4.881499435
conjunctions_total	**4.850229186**
ttrr	4.744063714
nouns_total	**4.63369171**
unique_words	4.609612529

Feature	Score
pronouns_density	**3.860901774**
proportion_vocab_list[a1]_total_words	3.355255868
crawford	3.083923886
proportion_vocab_list[abnormal]_headwords	3.059688495
sentence_count	2.858495234
total_sentences	**2.780063655**
proportion_vocab_list[absentminded]_total_words	2.685789236
ttr_nouns	**2.41010852**
ts_avg_sent_per_word	2.248255026
proportion_vocab_list[abet]_total_words	2.182106948
proportion_vocab_list[abet]_headwords	2.182106948
proportion_vocab_list[abomasum]_total_words	2.182106948
proportion_vocab_list[abomasum]_headwords	2.182106948
pronouns_noun_ratio	**2.106469914**
sent_density	**2.048119793**
proportion_vocab_list[abate]_total_words	1.722627897
proportion_vocab_list[abate]_headwords	1.696145125
proportion_vocab_list[a1]_headwords	1.614302704
nouns_unique	**1.429434557**

Table E.5: Coherence Feature Importance Tests

Overall

What's the overall category and am yeah I can't simply categories performances for you don't insights into the features it's it's using versus sun features and it's not using because consists of all features however you that I see here is in narrowing down and the features for this category for you yeah as in at the moment it is using all 57 of them and and and maybe more only 19.

Feature	Score	Feature	Score	Feature	Score
proportion_vocab_list[abandon]_headwords	14.27582377	lexicon_count	6.93800963	proportion_vocab_list[abet]_total_words	2.77008194
difficult_words	13.41906313	total_words	6.834380155	proportion_vocab_list[abet]_headwords	2.77008194
polysyllabcount	13.29707246	nlemmas	6.773600167	proportion_vocab_list[abomasum]_total_words	2.77008194
proportion_vocab_list[accent]_headwords	11.3781753	n_bigram_lemmas	6.773600167	proportion_vocab_list[abomasum]_headwords	2.77008194
grammar_errors_ltp_by_words	10.02287274	n_trigram_lemmas	6.773600167	proportion_vocab_list[abandon]_total_words	2.560321881
syllable_count	9.680129981	proportion_vocab_list[abaft]_headwords	6.175245252	determiners_per_words	2.497674073
pct_transitions	9.2517999	proportion_vocab_list[abaft]_total_words	6.144068368	nouns_total	2.348175159
n_trigram_lemma_types	8.952429235	ncontent_tokens	5.601640629	nouns_unique	2.180390661
num_types	8.952429235	function_ttr	5.590956532	proportion_vocab_list[abnormal]_total_words	2.177297632
nlemma_types	8.952429235	proportion_vocab_list[abnormal]_headwords	5.37E+00	proportion_vocab_list[absentminded]_total_words	2.163955982
n_bigram_lemma_types	8.952429235	proportion_vocab_list[a]_headwords	5.094092462	conjunctions_total	1.943824923
determiners_total	8.946634884	pronouns_total	4.941964937	proportion_vocab_list[accent]_total_words	1.924996303
letter_count	8.360611133	monosyllabcount	4.86820757	proportion_vocab_list[abashed]_total_words	1.921267615
ncontent_types	8.261155741	total_sentences	4.242184438	proportion_vocab_list[abashed]_headwords	1.921267615
reading_time	8.192861291	sentence_count	4.196859193	proportion_vocab_list[aback]_total_words	1.477536266
char_count	8.191934694	proportion_vocab_list[a]_total_words	4.152176495	pronouns_density	1.201514767
nfunction_tokens	7.81515932	crawford	3.163807878	proportion_vocab_list[abate]_total_words	1.083243997
pct_rel_trigrams	7.552931616	text_standard	3.063653346	proportion_vocab_list[aback]_headwords	1.052754839
nfunction_types	7.307528473	smog_index	3.025845124	linsear_write_formula	1.04153577

Table E.6: Overall Feature Importance Tests

E.2 The Predictor

To experiment with The Predictor's understanding of the relationship between an essay's features and their respective computer-generated scores. I have explored multiple models, in conjunction with multiple datasets, and performed a Train-Test Evaluation for each. My motivation for these was to not only observe the performance of the predictor for the domain

data. But also to gain insight into its relative performance to other systems, and observe its robotness with similar data.

Test Train Split This process is used to evaluate the performance of a machine learning algorithm. It requires data as an input which demonstrates the relationship in which we're trying to synthesize [x]. So in our case, this data is the numerical representations of the features from real world essays along with their respective human-given scores from the EPP. Then this data is divided into two subsets. The first is provided to the model for it to configure it's understanding of this relationship. The second is used to test it's understanding of this relationship. These sets must be different because the underlying purpose of this type of investigation - to observe a model's ability to predict data that it hasn't seen before [x]. Based on it's base-nature combined with it's configured understanding of this relationship which has been refined using the train dataset. Once the test dataset has been run through the model, we can then simply compare the model's predicted scores with the actual scores to produce a number of metrics to explain the model's generalized prediction performance.

Considerations In general, this procedure is appropriate when the dataset is a suitable representation of the domain and is an appropriate size. If both of these criteria are not met, less optimal results can occur. The main concern with the Hewlett Datasets is their ability to accurately represent the domain problem. This is discussed in the datasets section. In terms of the VUW dataset, it is on the small side. The optimal size for a machine learning algorithm can vary based on the problem. It can require thousands or millions of examples. What I've observed in other AES studies is between 1000-2000 essays [17]. However the EPP data only consists of 113 essays. This is a concern as it can mean that the data from this set can be poorly descriptive of the relationship under investigation. As it may not exhibit a representative distribution of the common and uncommon cases that make up the general relationship of the data. Which can mean that the model can receive insufficient resources to configure an effective understanding between the inputs and outputs. This can also mean that the error measures can be overly accurate [x]. A suitable alternative procedure in this case is the k-fold cross-validation [x]. However this was not done in this study. The error measures are the specific calculations that we use to interpret the performance of the model.

Agreement Tests In statistics, inter-rater reliability (also called by various similar names, such as inter-rater agreement, inter-rater concordance, inter-observer reliability, and so on) is the degree of agreement among independent observers who rate, code, or assess the same phenomenon. In this I propose four Agreement Measures measures to analyze the results; Quadratic Weighted Kappa (QWK), Pearson's Correlation (P), Adjacent Agreement (A), percentage, Exact Agreement percentage (E)

Quadratic Weighted Kappa Quadratic Weighted Kappa measures the agreement percentage between the machine and human after correcting for the likelihood that some agreement between raters might happen by random chance. Quadratic Weighted Kappa is considered to be the best measure of agreement in the AES field because of the chance agreement correction [18]. The QWK score is a ratio between -1 and 1. A negative QWK score implies that the agreement is "worse than random". A random model should give a score of close to 0. A perfect predictions will give a score of 1.0 [19]. Below are some guide lines for this metric proposed by Viera Garrett [20]. Additionally, the ETS uses a QWK value of 0.70 as an established baseline for the assessment of the level of agreement between a human and an AES

system. Anything less than this threshold is considered as a bad agreement and anything above it is considered as a good agreement[15].

QWK	Agreement Representation
0.01–0.20	slight agreement
0.21–0.40	fair agreement
0.41–0.60	moderate agreement
0.61–0.80	substantial agreement
0.81–0.99	almost perfect agreement

Pearson's Correlation Pearson's correlation is the product-moment correlation between the machine-produced predicted score and the human marker final score. It ranges from -1 to +1. A higher positive correlation indicates stronger agreement.

Adjacent Agreement Percentage Adjacent-agreement percentage refers to the agreement between scores (i.e., machine and human) that are within one point of one another, as a percentage. A value of 1.0 is perfect agreement and 0.0 is no agreement.

Exact Agreement Percentage Exact-agreement percentage refers to the agreement between two scores, as a percentage. A value of 1.0 is perfect agreement and 0.0 is no agreement.

E.2.1 Overall

E.2.2 VUW EPP

Score	Category	QWK	P	A	E	Model
Grammar	**Grammar**	**0.295**	**0.312**	**0.957**	**0.652**	**RFR**
Grammar	Vocab	0.253	0.265	0.913	0.478	DTR
Grammar	Flow	0.359	0.362	1	0.522	KR
Grammar	Ideas	0.064	0.07	0.87	0.348	DTR
Grammar	Coherence	0.233	0.258	0.957	0.522	GBR
Grammar	Overall	0.19	0.211	0.957	0.565	EN
Vocab	Grammar	0.299	0.406	0.957	0.609	GBR
Vocab	**Vocab**	**0.476**	**0.544**	**1**	**0.609**	**GBR**
Vocab	Flow	0.343	0.359	1	0.565	KR
Vocab	Ideas	0.189	0.35	0.957	0.435	LR
Vocab	Coherence	0.343	0.467	0.957	0.565	LR
Vocab	Overall	0.499	0.509	1	0.609	DTR
Flow	Grammar	0.373	0.5	1	0.435	LR
Flow	Vocab	0.389	0.398	1	0.478	DTR
Flow	**Flow**	**0.405**	**0.41**	**1**	**0.522**	**DTR**
Flow	Ideas	0.361	0.361	1	0.565	DTR
Flow	Coherence	0.28	0.289	0.957	0.478	DTR
Flow	Overall	0.287	0.339	1	0.522	KR

Score	Category	QWK	P	A	E	Model
Ideas	Grammar	0.42	0.527	1	0.478	SVR
Ideas	Vocab	0.41	0.417	0.957	0.478	DTR
Ideas	Flow	0.386	0.507	1	0.435	SVR
Ideas	**Ideas**	**0.42**	**0.527**	**1**	**0.478**	**SVR**
Ideas	Coherence	0.383	0.451	0.957	0.522	LR
Ideas	Overall	0.386	0.507	1	0.435	SVR
Coherence	Grammar	0.265	0.352	0.957	0.522	SVR
Coherence	Vocab	0.43	0.442	0.957	0.565	DTR
Coherence	Flow	0.426	0.436	0.957	0.652	DTR
Coherence	Ideas	0.265	0.352	0.957	0.522	SVR
Coherence	**Coherence**	**0.419**	**0.515**	**1**	**0.565**	**SVR**
Coherence	Overall	0.332	0.361	1	0.522	KR
Overall	Grammar	0.359	0.422	1	0.522	RFR
Overall	Vocab	0.515	0.517	1	0.565	DTR
Overall	Flow	0.437	0.443	1	0.478	DTR
Overall	Ideas	0.337	0.427	1	0.391	GBR
Overall	Coherence	0.405	0.516	1	0.522	LR
Overall	**Overall**	**0.535**	**0.562**	**1**	**0.652**	**KR**

Table E.7: Categorical & Overall Agreement Measures

Categorical

For the Grammar category, the QWK was 0.295 which represents "fair agreement" between the machine predicted essay score and the human marker essay score. The correlation between the machine predicted score and the human-produced score was 0.312 which indicates a weak correlation. The adjacent agreement was 0.957 meaning that the machine

and human score were within one point of one another 95% of the time. Exact agreement between the machine-predicted score and the human-produced score occurred 65% of the time. The highest scoring model was the RFR.

For the Vocab category, the QWK was 0.476 which represents "moderate agreement" between the machine predicted essay score and the human marker essay score. The correlation between the machine predicted score and the human-produced score was 0.544 which indicates a moderate correlation. The adjacent agreement was 1 meaning that the machine and human score were within one point of one another 100% of the time. Exact agreement between the machine-predicted score and the human-produced score occurred 60% of the time. The highest scoring model was the GBR.

For the Flow category, the QWK was 0.405 which represents "fair agreement" between the machine predicted essay score and the human marker essay score. The correlation between the machine predicted score and the human-produced score was 0.41 which indicates a moderate correlation. The adjacent agreement was1 meaning that the machine and human score were within one point of one another 100% of the time. Exact agreement between the machine-predicted score and the human-produced score occurred 52% of the time. The highest scoring model was the DTR.

For the Ideas category, the QWK was 0.42 which represents "moderate agreement" between the machine predicted essay score and the human marker essay score. The correlation between the machine predicted score and the human-produced score was 0.527 which indicates a moderate correlation. The adjacent agreement was 1 meaning that the machine and human score were within one point of one another 100% of the time. Exact agreement between the machine-predicted score and the human-produced score occurred 47% of the time. The highest scoring model was the SVR.

Overall

For the Overall category, the QWK was 0.535 which represents "moderate agreement" between the machine predicted essay score and the human marker essay score. The correlation between the machine predicted score and the human-produced score was 0.562 which indicates a moderate correlation. The adjacent agreement was 1 meaning that the machine and human score were within one point of one another 100% of the time. Exact agreement between the machine-predicted score and the human-produced score occurred 65% of the time. The highest scoring model was the KR.

E.2.3 Hewlett

Overall

Dataset	Score	Category	QWK	P	A	E	Model
1	Overall	Overall	0.798	0.84	0.899	0.459	RFR
3	Overall	Overall	0.546	0.654	0.977	0.471	BR
4	Overall	Overall	0.634	0.744	0.977	0.46	SVR
5	Overall	Overall	0.735	0.736	0.967	0.587	DTR
6	Overall	Overall	0.647	0.739	0.969	0.458	LR
7	Overall	Overall	0.747	0.761	0.392	0.131	KR
8	Overall	Overall	0.711	0.753	0.393	0.152	GBR

Table E.8: trtgfhfgh

For the Overall Scores for all Hewlett datasets, the QWK was between 0.546-798. For all most of all of these, the value was above 0.634 which represents "substantial agreement" between the machine predicted essay score and the human marker essay score. The correlation between the machine predicted score and the human-produced score was mostly within the 0.7 range which indicates a strong correlation. The adjacent agreement was very close to 1 for all datasets except for the last two meaning that the machine and human score were within one point of one another almost 100% of the time. For the datasets which did not have as measure here. This is likely due to their different scoring ranges. As the overall score ranges for these datasets are ... and ... . Exact agreement between the machine-predicted score and the human-produced score occurred approximately on 50% of the time for all datasets except the last two. Again, I believe this is due to their score range differences. The highest scoring models were quite varied: RFR, BR, SVR, DTR, LR, KR, GBR.

Categorical

Score	Category	QWK	P	A	E	Model
Writing_Applications	Grammar	0.565	0.684	0.986	0.489	SVR
Writing_Applications	Vocab	0.583	0.585	0.986	0.581	DTR
Writing_Applications	Flow	0.596	0.6	0.981	0.567	DTR
Writing_Applications	Ideas	0.567	0.687	0.986	0.483	SVR
Writing_Applications	Coherence	0.558	0.56	0.975	0.586	DTR
Writing_Applications	Overall	0.617	0.621	0.983	0.597	DTR
Language_Conventions	Grammar	0.626	0.629	0.983	0.653	DTR
Language_Conventions	Vocab	0.544	0.549	0.975	0.561	DTR
Language_Conventions	Flow	0.482	0.488	0.975	0.519	DTR
Language_Conventions	Ideas	0.342	0.529	0.969	0.347	SVR
Language_Conventions	Coherence	0.404	0.409	0.95	0.528	DTR
Language_Conventions	Overall	0.566	0.571	0.983	0.586	DTR

Table E.9: Hewlett Set #2 Agreement Measures

ll

For the Ideas category, the QWK was 0.425 which represents "moderate agreement" between the machine predicted essay score and the human marker essay score. The correlation between the machine predicted score and the human-produced score was 0.505 which indicates a moderate correlation. The adjacent agreement was very close to one meaning that the machine and human score were within one point of one another almost 100% of the time. Exact agreement between the machine-predicted score and the human-produced score occurred 36% of the time. The highest scoring model was the SVR. These measures are not that strong although this is to be expected as the ideas category was the most underdeveloped feature category. It only consists of three very simple features.

Score	Category	QWK	P	A	E	Model
Overall	Grammar	0.696	0.713	0.341	0.118	RFR
Overall	Vocab	0.719	0.741	0.341	0.118	GBR
Overall	Flow	0.708	0.725	0.369	0.127	GBR
Overall	Ideas	0.706	0.726	0.347	0.124	GBR
Overall	Coherence	0.747	0.761	0.366	0.14	LR
Overall	Overall	0.747	0.761	0.392	0.131	KR
Ideas	Grammar	0.427	0.429	0.908	0.468	DTR
Ideas	Vocab	0.427	0.545	0.952	0.341	LR
Ideas	Flow	0.425	0.559	0.946	0.35	GBR
Ideas	Ideas	0.425	0.505	0.962	0.36	SVR
Ideas	Coherence	0.46	0.552	0.968	0.398	SVR
Ideas	Overall	0.475	0.607	0.939	0.369	LR
Organization	Grammar	0.44	0.441	0.968	0.564	DTR
Organization	Vocab	0.43	0.43	0.965	0.545	DTR
Organization	Flow	0.445	0.445	0.959	0.567	DTR
Organization	Ideas	0.491	0.491	0.981	0.554	DTR
Organization	Coherence	0.466	0.596	0.984	0.465	BR
Organization	Overall	0.456	0.587	0.984	0.452	KR
Style	Grammar	0.458	0.459	0.987	0.637	DTR
Style	Vocab	0.474	0.476	0.994	0.64	DTR
Style	Flow	0.421	0.422	0.987	0.627	DTR
Style	Ideas	0.507	0.508	1.0	0.643	DTR
Style	Coherence	0.479	0.48	0.997	0.621	DTR
Style	Overall	0.469	0.47	0.997	0.624	DTR
Conventions	Grammar	0.417	0.418	0.968	0.599	DTR
Conventions	Vocab	0.407	0.408	0.981	0.567	DTR
Conventions	Flow	0.375	0.376	0.975	0.541	DTR
Conventions	Ideas	0.392	0.393	0.975	0.567	DTR
Conventions	Coherence	0.439	0.439	0.971	0.618	DTR
Conventions	Overall	0.418	0.541	0.994	0.522	BR

Table E.10: Hewlett Set #7 Agreement Measures

Score	Category	QWK	P	A	E	Model
Overall	Grammar	0.657	0.702	0.338	0.124	GBR
Overall	Vocab	0.689	0.724	0.324	0.152	GBR
Overall	Flow	0.618	0.648	0.29	0.09	RFR
Overall	Ideas	0.541	0.574	0.228	0.076	GBR
Overall	Coherence	0.656	0.694	0.29	0.09	LR
Overall	Overall	0.711	0.753	0.393	0.152	GBR
Ideas_And_Content	Grammar	0.439	0.539	0.972	0.566	SVR
Ideas_And_Content	Vocab	0.432	0.603	0.979	0.455	RFR
Ideas_And_Content	Flow	0.454	0.568	0.979	0.559	SVR
Ideas_And_Content	Ideas	0.443	0.54	0.972	0.572	SVR
Ideas_And_Content	Coherence	0.428	0.566	0.972	0.503	RFR
Ideas_And_Content	Overall	0.493	0.495	0.979	0.552	DTR
Organization	Grammar	0.485	0.58	0.993	0.566	SVR
Organization	Vocab	0.528	0.529	0.979	0.662	DTR
Organization	Flow	0.503	0.611	0.993	0.572	SVR
Organization	Ideas	0.493	0.583	0.993	0.579	SVR
Organization	Coherence	0.45	0.451	0.986	0.586	DTR
Organization	Overall	0.479	0.603	0.986	0.552	SVR
Voice	Grammar	0.372	0.515	0.972	0.593	LR
Voice	Vocab	0.456	0.461	0.986	0.607	DTR
Voice	Flow	0.385	0.526	0.979	0.628	SVR
Voice	Ideas	0.369	0.504	0.979	0.628	SVR
Voice	Coherence	0.402	0.404	0.972	0.621	DTR
Voice	Overall	0.411	0.412	0.979	0.607	DTR
Word_Choice	Grammar	0.451	0.461	1.0	0.634	DTR
Word_Choice	Vocab	0.41	0.415	0.993	0.69	DTR
Word_Choice	Flow	0.37	0.479	1.0	0.6	SVR
Word_Choice	Ideas	0.373	0.48	1.0	0.607	SVR
Word_Choice	Coherence	0.346	0.513	0.993	0.497	SVR
Word_Choice	Overall	0.409	0.424	0.993	0.669	DTR
Sentence_Fluency	Grammar	0.377	0.461	0.993	0.572	SVR
Sentence_Fluency	Vocab	0.416	0.424	0.972	0.634	DTR
Sentence_Fluency	Flow	0.366	0.462	0.986	0.545	SVR
Sentence_Fluency	Ideas	0.377	0.461	0.993	0.572	SVR
Sentence_Fluency	Coherence	0.342	0.53	0.979	0.421	SVR
Sentence_Fluency	Overall	0.37	0.377	0.979	0.566	DTR
Conventions	Grammar	0.348	0.348	0.993	0.503	DTR
Conventions	Vocab	0.379	0.384	0.979	0.517	DTR
Conventions	Flow	0.35	0.364	0.972	0.503	DTR
Conventions	Ideas	0.376	0.381	0.979	0.497	DTR
Conventions	Coherence	0.404	0.406	0.993	0.552	DTR
Conventions	Overall	0.411	0.422	0.979	0.641	DTR

Table E.11: Hewlett Set #8 Agreement Measures

Appendix F

Conclusion

F.1 Current State

Accuracy The current state of the system is that it seems to perform really well when predicting the overall score, for the, much larger, Hewlett Kaggle Competition Data. However, in all other cases, the results are just not strong enough for this system to be released yet. Fortunately, there are still a number of refinements that can be performed as well as further feature discovery.

Easy of Use, Explain-ability In terms of the systems easy of use and explain-ability, because these traits have been heavily prioritised throughout the project, their overall nature is strong.

F.2 Future Work

The Feature Extractor One observation that stood out to me was the overlapping of common features within the top Features for a Feature Category. Going forward, I think the first most optimal task to improve performance, would be to firstly, update The Feature Extractor implementation so that it can process the same feature to multiple categories without any bugs.

Feature Categories I have observed good results for both data sets for using all features to predict the overall scores for these datasets. However, the system performs relatively worse when trying to predict a specific marking category. As well as this, in the feature importance tests, it was clear that all feature categories are not incorporating most of their strongest features. Both of these traits lead me to believe that the feature's for each category can be further refined.

Following this, I would update the feature categories according to the feature importance test. Then, to observe, the change in performance I would execute a batch job for all datasets against The Feature Extractor and Agreement Calculators.

Sample Size Although it is not relatively surprising to observe a performance decrease with the domain data. I think it would be beneficial to understand that relationship netter. So another area which would be good to gain clarity into would be the relationship between the sample size and the performance with the EPP VUW data.

[1]"English for Academic Purposes," Victoria University of Wellington, 2021. https://www.wgtn.ac.nz/equals/english-academic-purposes/overview (accessed Oct. 31, 2021).

[2]M. D. Shermis and J. C. Burstein, Automated Essay Scoring: A Cross-disciplinary Perspective. Routledge, 2003.

[3]A. Harris et al., "A Study of the Use of the e-rater Scoring Engine for the Analytical Writing Measure of the GRE revised General Test," Educational Testing Service, 2014.

[4]M. D. Shermis and J. Burstein, Handbook of Automated Essay Evaluation: Current Applications and New Directions. Routledge, 2013.

[5]J. H. Friedman, "Recent Advances in Predictive (Machine) Learning," Journal of Classification, vol. 23, no. 2, pp. 175–197, Sep. 2006, doi: 10.1007/s00357-006-0012-4.

[6]Contributors to Wikimedia projects, "Mathematical model," Wikipedia, Oct. 06, 2021. https://en.wikipedia.org/wiki/Mathematical_$model(accessedOct.31,2021)$.

[7]Contributors to Wikimedia projects, "Regression analysis," Wikipedia, Oct. 02, 2021. https://en.wikipedia.org/wiki/Regression_$analysis(accessedOct.31,2021)$.

[8]Contributors to Wikimedia projects, "Statistical classification," Wikipedia, Jun. 11, 2021. https://en.wikipedia.org/wiki/Statistical_$classification(accessed2021)$.

[9]Contributors to Wikimedia projects, "Artificial neural network," Wikipedia, Oct. 29, 2021. https://en.wikipedia.org/wiki/Artificial_$neural_network(accessedOct.31,2021)$.

[10]Contributors to Wikimedia projects, "Feature extraction," Wikipedia, Oct. 20, 2021. https://en.wikipedia.org/wiki/Feature_$extraction(accessedOct.31,2021)$.

[11]M. Uto, Y. Xie, and M. Ueno, "Neural Automated Essay Scoring Incorporating Handcrafted Features," 2020. Accessed: Oct. 31, 2021. [Online]. Available: http://dx.doi.org/10.18653/v1/2020.coling-main.535

[12]H. Chen and B. He, "Automated Essay Scoring by Maximizing Human-machine Agreement," presented at the Conference on Empirical Methods in Natural Language Processing, Seattle, Washington, USA, 2013. Accessed: 2021. [Online]. Available: http://citeseerx.ist.psu.edu/vi

[13]D. Pyle, Business Modeling and Data Mining. Elsevier, 2003.

[14]Contributors to Wikimedia projects, "Nonlinear system," Wikipedia, Sep. 24, 2021. https://en.wikipedia.org/wiki/Nonlinear_$system(accessedOct.31,2021)$.

[15]M. Lilja, "Automatic Essay Scoring of Swedish Essays using Neural Networks," Thesis, Department of Statistics Uppsala University, 2018. Accessed: 2021. [Online]. Available: https://uu.diva-portal.org/smash/get/diva2:1213688/FULLTEXT01.pdf

[16]Victoria University of Wellington, "Vocabulary Lists," Victoria University of Wellington, 2021. https://www.wgtn.ac.nz/lals/resources/paul-nations-resources/vocabulary-lists (accessed 2021).

[17]A. T. Moore, "Applying the Developmental Path of English Negation to the Automated Scoring of Learner Essays ," Thesis, Brigham Young University, 2018. Accessed: 2021. [Online]. Available: https://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=7835context=etd

[18]J. Shin and M. Gierl, "Using Automated Procedures to Machine Score Essays Written in the New Zealand NCEA Examination Content Areas of English and History," Centre for Research in Applied Measurement and Evaluation, University of Alberta, Jun. 2020. [Online]. Available: https://www.nzqa.govt.nz/assets/About-us/Future-State/NCEA-Online/research-and-innovation/UoA-CRAME-report-Automated-Essay-Marking.pdf

[19]carlolepelaars, "Understanding The Metric: Quadratic Weighted Kappa," Kaggle, Nov. 23, 2019. Accessed: Nov. 05, 2021. [Online]. Available: https://www.kaggle.com/carlolepelaars/understanding-the-metric-quadratic-weighted-kappa

[20]A. Viera and J. Garrett, "Understanding interobserver agreement: the kappa statistic," Fam med, 2005.