MASTER THESIS

AUTOMATIC ESSAY SCORING:
MACHINE LEARNING MEETS APPLIED LINGUISTICS

Victor Dias de Oliveira Santos

July, 2011

UNIVERSITÄT
DES
SAARLANDES

European Masters in Language and Communication Technologies

Supervisors:

Prof. John Nerbonne

Prof.Marjolijn Verspoor

Rijksuniversiteit Groningen / University of Groningen

Co-supervisor:

Prof. Manfred Pinkal

Universität des Saarlandes / University of Saarland

Declaration of the author

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

---------- ----------

Signature: Victor D.O. Santos Date

Abstract

Automated Essay Scoring (AES) has for quite a few years now attracted substantial attention from government, language researchers and others interested in automatically assessing language proficiency. Sometimes the task is tackled by focusing on many variables (many of which are not relevant for the construct at hand) and sometimes by focusing on few (there are even cases of univariate analysis). However, typical real-word data includes various attributes, only a few of which are actually relevant to the true target concept (Landwehr, Hall, & Frank, 2005). In this Master thesis, we investigate several machine learning algorithms which are part of the widely used WEKA package (University of Waikato) for data mining and analyze them not only in terms of how well they perform with regard to their accuracy in assessing essays in English manually annotated for more than 81 features, but also with regard to how they can be said to reflect research findings in Applied Linguistics. Some models, such as Logistic Model Tree (LMT) achieve better accuracy than others and expose the variables that correlate the most with proficiency level and which function most importantly in classification. We also explore the importance of feature selection for improving classifiers and to what extent automatic essay scoring systems and human raters might be said to differ in their scoring procedures. Finally, we explore how the variables that have been found to correlate the most with proficiency level can be implemented in an automatic system. The dataset used in our experiments comes from English essays written by Dutch students and collected within the framework of the OTTO project, which is financed by the OCW (Dutch Ministry or Education), European Platform and Network of Bilingual schools.

Acknowledgment

First, I would like to express my gratitude and thanks to my thesis supervisors: John Nerbonne (University of Groningen), Marjolijn Verspoor (University of Groningen) and Manfred Pinkal (University of Saarland). Thanks for taking the time to answer the sometimes overwhelming number of emails I would send on a single day and for our laid-back and very fruitful discussions and meetings. I have learned a lot with you. It has been a pleasure working under your supervision and I truly hope we can collaborate further sometime soon.

Secondly, I would like to thank my mother for her perfect mixture of unconditional love, support and wisdom to say the right thing at the right time (even if it might be hard to hear and swallow sometimes).

Thirdly, I would like to thank all the great friends I have made during this Master's program in Language and Communication Technology for their support and for all the good times we have enjoyed together, which I am sure have contributed to my success in the program. In special, I would like to thank my good friend and former LCT student Yevgeni Berzak for being such a wise person, for his support throughout the program and for his friendship. A special thanks goes to the local coordinator at the University of Groningen, Gosse Bouma, for his patience and easy-going attitude to problem solving and to Bobbye Pernice and Maria Jacob, at the University of Saarland, for always making things less complicated than they needed to be.

To an amazing woman and (my) mother,

Maria Elisa de Oliveira Santos

TABLE OF CONTENTS

INTRODUCTION

Automated Essay Scoring (AES) has for quite a few years attracted substantial attention from government, language researchers and other parties interested in automatically assessing language proficiency. One of the best known examples of Automated Essay Scoring is the system used in the TOEFL exam (Test of English as a Foreign Language), called E-rater. When it comes to AES, the task is sometimes tackled by focusing on many variables (many of which may not be relevant for the construct at hand) and sometimes by focusing on few (there even being cases of univariate analysis, in which a single feature/variable is used). However, typical real-word data includes various attributes, only a few of which are actually relevant to the true target concept (Landwehr, Hall, & Frank, 2005).

In this thesis, we investigate to what extent machine learning tools and techniques, such as those implemented in the widely used WEKA package (University of Waikato) can help us with our task at hand: classifying/scoring essays according to English proficiency level. We are also interested in how machine learning can help us make the task of automatic essay scoring more feasible, by investigating which features are more indicative of proficiency level and how they lend themselves to automatic, with a view to a truly automatic essay scoring system. Given that machine learning is quite fitting for dealing with a large number of features and optimal at finding hidden patterns in data, we want to explore how suitable these algorithms are for dealing with the delicate and multivariate reality of second language proficiency. We also investigate if and how the outputs of some classifiers might reflect findings and common practice in Applied Linguistics when it comes to proficiency level assessment. Finally, we explore whether there might be fundamental differences in how Automatic Scoring Systems and human raters differ in their scoring procedures.

Chapter 1 introduces Machine Learning to the reader. Chapter 2 is anoverview of what Decision Trees are, how they are built and optimized and includes a short description of each of the DT classifiers we have explored. In Chapter 3, we introduce Bayesian Classifiers and show how their probabilistic approach to classification differs from that used in Decision Trees. Chapter 4 introduces the reader to our language data (set of holistically scored essays, annotated for more than 80 features) and deals with the results of the classifiers in our essay-scoring task in terms of accuracy, adjacent classifications, errors, mean scores, and correlation coefficient with human raters. The best classifier for our task, namely, Logistic Model Tree, is also discussed in this chapter. In Chapter 5 we discuss how our approach and results relate to work and findings in both the Automatic Essay Scoring and the Second Language Development literatures. Finally, Chapter 6 summarizes our work and presents possible future endeavors.

1. MACHINE LEARNING

The Department of Engineering at Cambridge University defines machine learning as follows:

Machine learning is a multidisciplinary field of research focusing on the mathematical foundations and practical applications of systems that learn, reason and act. Machine learning underpins many modern technologies, such as speech recognition, robotics, Internet search, bioinformatics, and more generally the analysis and modeling of large complex data. Machine learning makes extensive use of computational and statistical methods, and takes inspiration from biological learning systems. [1]

It is important to add here that one of the tasks of machine learning is to find patterns in and make inferences based on unstructured data.

One of the traditional areas of application for machine learning is classification, which is precisely what we intend to do with our collection of essays. Based on our corpus of essays, we would like to have a system that is able to classify each essay into one of 6 possible levels (0-5) with regard to English proficiency. Two of the methods used in Machine Learning for classification are: supervised methods and unsupervised methods. In supervised methods, the system (classifier) has access to the class label of each data sample and takes the class into account when building a classifier, by looking at the specific characteristics (features and their corresponding values) of each class. In unsupervised methods, the system has no access to class labels and has somehow to infer what (and often how many) the real classes present in the data are. This can be done, for example, through clustering, that is, grouping together data samples which show similar patterns. Given that all the essays we use in our work have already been holistically scored by human raters (we know the proficiency level of each

[1]http://cbl.eng.cam.ac.uk/Public/MLG/

essay), we will make use only of supervised methods. The algorithms/classifiers used in machine learning belong to several distinct families, each one tackling problems in specific ways. The two families of classifiers that we will explore in this thesis are: Decision Trees and Bayesian classifiers. These will be explained in more detail in future sections. Given the large number of features annotated in each essay and the large number of essays themselves, machine learning (performed here by means of the WEKA software) seems perfect for our task at hand. In addition, we will seek classifiers which not only show good classification accuracy but which are also transparent, that is, easy to interpret in the sense of (applied) linguistics.

We now turn to Decision Tree schemes and explore what they are and how decision trees can be built and optimized. It is important that the reader understand this in order to see why DTs are suitable for our essay-scoring task.

2. DECISION TREES

In this section, we look closely at what decision trees are and how they can be used in order to assign proficiency level to each one of the essays in our corpus based on the value of each feature. Moreover, we explore how decision trees are built and how they can be optimized by presenting the decision tree schemes we have experimented with in the scope of our work.

2.1. Definition

Decision Trees (DTs) are a specific machine learning scheme which is guided by what is usually termed as a "divide and conquer" approach. The basic idea of this approach is the following: if we must deal with a problem which may be too hard to tackle in its entirety all at once, let us then break it down into various sub-problems/tasks (thus "dividing") and find a solution to each of these sub-problems, one at a time. In the end, we will end up with a solution to our original problem (thus "conquering").

In a classification problem, one is interested in assigning a class to a given input, based on the characteristics (attributes/features and their corresponding values) of that input. Classes (we will not deal with numeric classes in the examples below, but only with nominal/categorical ones) can come in basically an infinite number of shapes and colors, so to speak, as exemplified below:

a) *Yes or No* (in the case of deciding whether someone should be hired or not)
b) *German, Hungarian, Portuguese, Dutch, Spanish* (when trying to decide the language a document is written in, for example)
c) Zero, One, Two, Three, Four or Five (if trying to decide which level of English a certain student is at based on an essay they have written)

d) *Spam/Non-Spam* (when deciding whether a certain email is a spam or not).

e) and so forth.

In all these problems, the scenario is the same. We have a group of features and corresponding values that we must analyze in order to decide which class a given sample (be it an essay, some weather data or an email) belongs to, in opposition to all the other classes it does NOT belong to.

Within the family of classifiers we call Decision Trees; there are several possible implementations, each one with their own specificities and methods. Nevertheless, the "divide and conquer" approach defined above applies to all of them. We will briefly look at different implementations of DTs in section 2.6.

2.2 – The Basic Idea

Decision Trees are fairly simple to understand. They are basically a way of sorting data into different paths, each of which will eventually lead to a classification. The tree will look similar to a genealogical tree from a distance. Each node inherits all the attribute values of their ancestors. At each point/node in a decision tree (with the exception of leaves), a question (or a combination of questions) is asked and according to the answer, data samples are allocated to one path/branch or another of the tree. This way, we start with our complete collection of samples at the top node of the tree and from then on at each node in the tree only a subset of the samples will be allocated to a specific branch. This process continues until no more questions are asked (no more attributes/features are checked) and a final classification is made. In the next section we exemplify this process, called "divide and conquer", in more detail.

2.3 - "Divide and Conquer"

Every DT looks exactly the same at its root, that is, at its top-most node. A node in a DT, as mentioned above, is basically a point in the tree at which a decision

has to be made. The root node (from where the tree starts growing) contains all the samples that we need to classify. Consequently, this is the least informative point in the tree. From the root node, we must choose one attribute/variable to analyze in the samples in order to decide how to treat those samples from that point on (see the invented language identification example in Figure 1 below). We must therefore further grow the tree, creating branches that will leave the root node, each one associated with one specific value of the attribute/feature upon which they were created and containing a subset of the samples present at the root node.

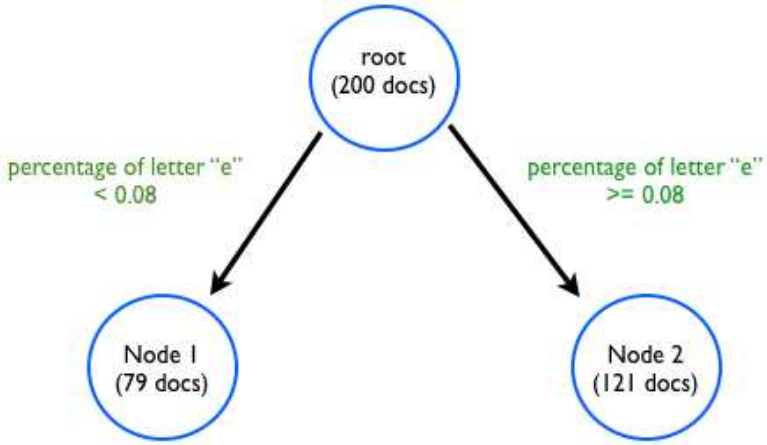

Figure 1 – *A possible language identification/classification task*

In our example above, after checking how often the letter "e" appears in each document, we are able to make an initial decision as to how to deal with a specific document from that point onwards. DTs have two types of nodes: internal nodes and leaf nodes. Internal nodes are nodes in the tree that have child nodes themselves, whereas leaf nodes are nodes that do not branch any further.

2.4 Building a Decision Tree

Before building a decision tree, all we have is a collection of items (samples) we want to infer patterns from and which will hopefully help us classify unseen data in the future. All these items are at a place in the tree that we call "the root node" (see previous section), since it is from this node that we will start growing our

tree. The standard procedure of building DTs is by checking among all possible attributes in our training set for the one that helps the most in reducing our uncertainty (also referred to as "entropy") as to which class a training sample belongs to and therefore helps to separate samples which are likely to belong together from those that are likely to be different.

We have chosen to use a traditional example in machine learning, namely "the weather problem", due to both its small number of attributes and to its intuitive understanding. It will help us with understanding the terminology needed. In this section and sections to follow, all tables and figures pertaining to the weather problem have been taken either from the book *Data Mining: Practical Machine Learning Tools and Techniques,* by Ian H. Witten & Eibe Frank (2005) or from running an analysis of the weather data in WEKA itself. The table below contains the data with respect to the weather problem:

Relation: weather

No.	outlook Nominal	temperature Numeric	humidity Numeric	windy Nominal	**play** Nominal
1	sunny	85.0	85.0	FALSE	no
2	sunny	80.0	90.0	TRUE	no
3	overcast	83.0	86.0	FALSE	yes
4	rainy	70.0	96.0	FALSE	yes
5	rainy	68.0	80.0	FALSE	yes
6	rainy	65.0	70.0	TRUE	no
7	overcast	64.0	65.0	TRUE	yes
8	sunny	72.0	95.0	FALSE	no
9	sunny	69.0	70.0	FALSE	yes
10	rainy	75.0	80.0	FALSE	yes
11	sunny	75.0	70.0	TRUE	yes
12	overcast	72.0	90.0	TRUE	yes
13	overcast	81.0	75.0	FALSE	yes
14	rainy	71.0	91.0	TRUE	no

Figure 2 -*Weather data (taken from WEKA)*

We have five variables and 14 instances (training samples) from which we have to build our DT (notice that this is fully supervised, since we know whether there will be a game or not). There are 4 predictor variables/attributes (*outlook,temperature*, *humidity* and *windy)*, which are used to help predict another variable, called the class variable(in our case, the variable *play*). Some of

the attributes are numeric (*temperature* and *humidity*), whereas others are nominal (*outlook, windy* and play). Numeric attributes (sometimes also loosely referred to as "continuous") have as values either integers or real numbers, whereas nominal attributes (also called categorical) have a small set of possible values.

For each node, we have to decide which attribute should be used to split it and also whether we should indeed split that specific node or simply turn it into a leaf node, at which a final classification will be made as to which class a sample that arrived at that node belongs to. The common ways of doing this are outlined in section 2.5. We can see below (Figure 3) a fully-grown tree for the *weather problem*:

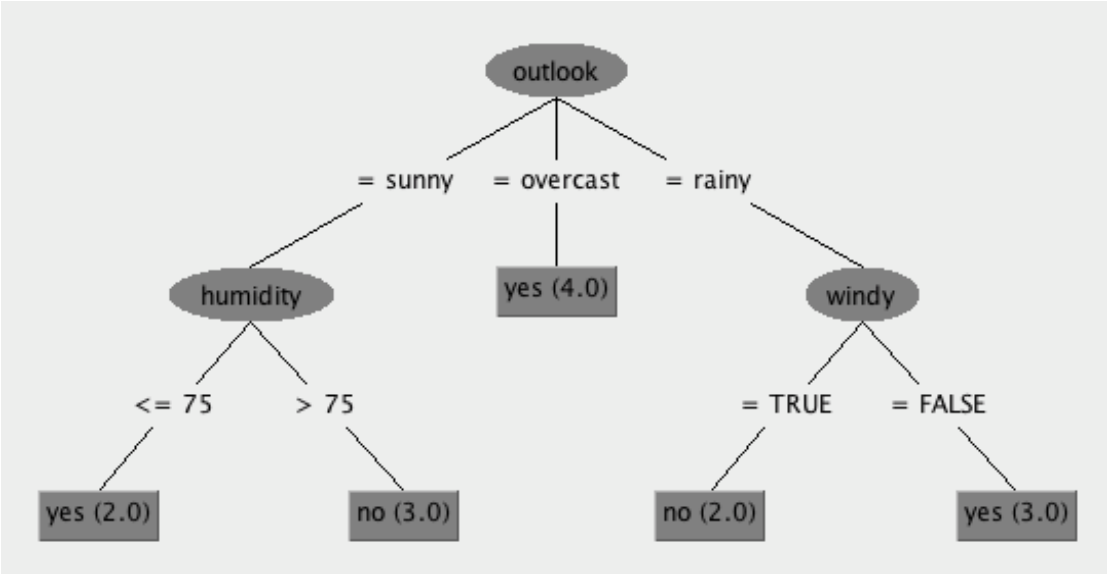

Figure 3 – *A possible DT for the weather data* (visualization in WEKA)

We now proceed to showing the two most commonly used measures in deciding which attribute to use for splitting a node, namely, Information Gain and the Gini Index. Due to a lack of space, we will not discuss other methods, such as Gain Ratio or Purity (how pure in terms of containing only one class a node is).

2.4.1 Information Gain

The notion of Information Gain (IG) is dependent on the more basic notion of *information* (or entropy). The information in a system can be said to be higher the more uncertainty there is in the system, that is, the more difficult it is to predict an outcome generated by the system. In a simple case, if we have 3 colored balls, for example, and each one is of a different color, our chances of guessing the color of a randomly drawn ball is about 33%. However, if we had 10 differently colored balls, our chances would be 10%. In this way, the second scenario/system is said to contain more information than the first. Information is usually calculated through a mathematical measure called *entropy* (the higher the entropy the higher the information and therefore the higher the uncertainty), represented by a capital (H). The formula for calculating entropy (whose result is usually given in bits due to the base of the log often being 2) is the following:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$$

It is important to note here that *P* is a probability distribution, in which the probabilities of each possible and discrete value P_i can take must add up to 1. Calculating the entropy at the root node of our weather problem, we get the following:

Entropy at root = - 5/14 * $\log_2$ 5/14 – 9/14 *$\log_2$9/14 = **0.940 bits**

We are now ready to calculate Information Gain for each attribute on which we might consider splitting a certain node. The basic idea behind it is to compare how much reduction in entropy/information each attribute is able to provide for our data and pick the one that provides the most reduction. We calculate IG for each possible attribute with relation to a specific node in the following manner, with the index *i* iterating over the child nodes of the current node:

$$\textbf{IG (attribute}_x\textbf{) = entropy (current\_node) } - \sum_{i=1}^{n} \textbf{P (child\_node)}_i \textbf{ * entropy (child\_node)}_i$$

Splitting on the attribute "outlook", for example, at our root node, gives us the outcome shown in Figure 4:

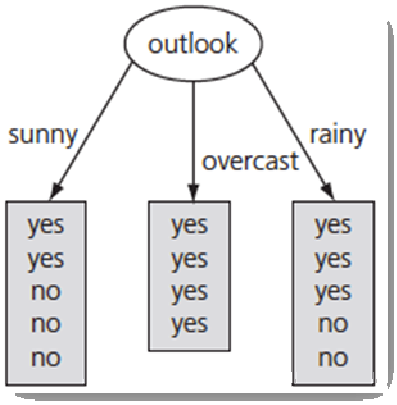

Figure 4: *First split on weather data*
(taken from 'Data Mining Practical Machine Learning Tools and Techniques')

The IG for attribute "outlook" in our weather problem is therefore:

IG (outlook) = info [5,9] – info [2,3], [4,0], [3,2] =
IG (outlook) = 0.940 – [5/14 * 0.971 + 4/14 * 0 + 5/14 * 0.971] =
0.940 – 0.693 = **0.247 bits**

If we calculate the IG for the other 3 attributes as well, we get:

IG (temperature) = 0.029 bits
IG (windy) = 0.048 bits
IG (humidity) = 0.152 bits

Given that we are interested in choosing the attribute that leads to a maximum increase in Information Gain, we decide therefore to split on the attribute

outlook at the root node. We do this recursively for nodes created subsequently, and no descendent nodes of a node should be split on a nominal attribute already used further above in its path. With numerical attributes, this is fine. As we will shortly explore (section 2.5), DTs usually stop growing either when we run out of attributes to split on or when we decide that a certain node should not be split any further (this might be done during the training phase or based on a development set, after the tree has first been fully grown). In section 2.5 we also discuss two possible ways of pruning decision trees, that is, making them smaller and less overfit for training data, namely *tree raising* and *tree substitution*.

2.4.2 Gini Index

Another common method for deciding on which attribute to split a node is called *Gini Index* (referred to as only *Gini* from now on), whose formula for a given node N is the following:

$$\mathbf{Gini(N) = 1 - (P_1^2 + P_2^2 + P_3^2 + \ldots + P_n^2)}$$

where $P_1 \ldots P_n$ are the relative frequencies of classes P_1 to P_n present at the node

Calculating the Gini at our root node, we have:

$$\mathbf{Gini\ (root)} = 1 - (5/14\ ^2 + 9/14\ ^2) =$$
$$1 - (0.413 + 0.127\ ) = \mathbf{0.459}$$

We then calculate the Gini for each possible attribute with relation to a specific node in the following manner:

$$\mathbf{Gini\ (attribute_x) = \sum_{i=1}^{n} P\ (child\_node)_i * Gini\ (child\_node)_i}$$

Splitting on the attribute *outlook,* for example, at our root node, gives us then the following Gini value for this split:

Gini (outlook) = 5/14 * Gini (sunny) + 4/14 * Gini (overcast) + 5/14 * Gini (rainy) = 5/14 * $[1-(2/5)^2+ (3/5)^2]$ + 4/14 * $[1-(4/4)^2]$ + 5/14 * $[1-(2/5)^2+ (3/5)^2]$

= 5/14 * [1 - 0.376] + 4/14 * 0 + 5/14 * [1 - 0.376]

= 2 * (5/14 * 0.624)

= 0.446

Calculating the Gini for attributes such as *humidity* and *temperature* is a little trickier in our case, given that these are not nominal attributes (in contrast to *outlook* or *windy*), but numerical ones. Numerical attributes need first to be discretized (grouped into a limited number of intervals) before being used in a task such as calculating the Gini. The typical way to discretize numeric attributes is by grouping the neighboring values together into interval groups in a way that we maximize the presence of a majority class in each of the groups. Due to the scope of this thesis, however, we will not get into the details of discretization and refer the reader to the book *Data Mining – Practical Machine Learning Tools and Techniques* (Witten & Frank, 2005) instead. We will use here a nominal version of the data (Figure 5) in order to calculate the Gini for the attributes *windy*, *temperature* and *humidity*:

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Figure 5 – *Weather data (nominal version, taken from WEKA)*

Gini (humidity) =7/14 * Gini (high) + 7/14 * Gini (normal)
= 7/14 * [1-(3/7)^2+ (4/7)^2] + 7/14 * [1-(6/7)^2+ (1/7)^2]
= 0.24489796 + 0.12244898
= 0.367

Gini (windy) = 8/14 * Gini (false) + 6/14 * Gini (true)
= 8/14 * [1-(6/8)^2+ (2/8)^2] + 6/14 * [1-(3/6)^2+ (3/6)^2]
= 0.214285… + 0.214285
= 0.428

Gini (temperature) = 4/14 * Gini (cool) + 4/14 * Gini (hot) + 6/14 * Gini (mild)
= 4/14 * [1-(3/4)^2+ (1/4)^2] + 4/14 * [1-(2/4)^2+ (2/4)^2] + 6/14 * [1-(4/6)^2+ (2/6)^2]
= 0.1071… + 0.1428… + 0.1904…
= 0.4403

Since we are interested in minimizing the Gini, we will choose the attribute *humidity* to split the root node. As we can see, Information Gain and Gini lead to different choices of attributes. This is due to the fact that both measurements have their specificities: IG is biased towards attributes with a large number of values and Gini prefers splits that lead to maximizing the presence of a single class after the split. Which one will turn out to be best will depend on the results on a test set.

2.5 Optimizing Decision Trees

A common practice in building Decision Trees is to first fully grow the tree (so that each leaf only contains samples belonging to one class) and then modify it. The inherent problem in using a fully-grown tree in a test set is that the model that has been built during the training phrase might, despite having very good classification performance on the training data, show poor classification results on the test set. This is due to the fact that the decision tree built might overfit the

training data and be therefore too specific, that is, customized to the training set. Decision Trees that accept some degree of impurity in their leaves usually do better when applied to new data. Modifying the fully grown tree so that it becomes more suitable for classifying new data is called post-pruning and usually consists of one (or both) of the following operations: subtree replacement and subtree raising.

2.5.1 Subtree replacement

Subtree replacement involves eliminating internal nodes of part of a tree (subtree) and replacing them by a leaf node found at the bottom of the subtree being eliminated. Figure 6 below, which represents labor negotiations in Canada, clarifies the idea. The label “good” indicates that both labor and management agreed on a specific contract. The label ”bad” indicates that no agreement was reached.

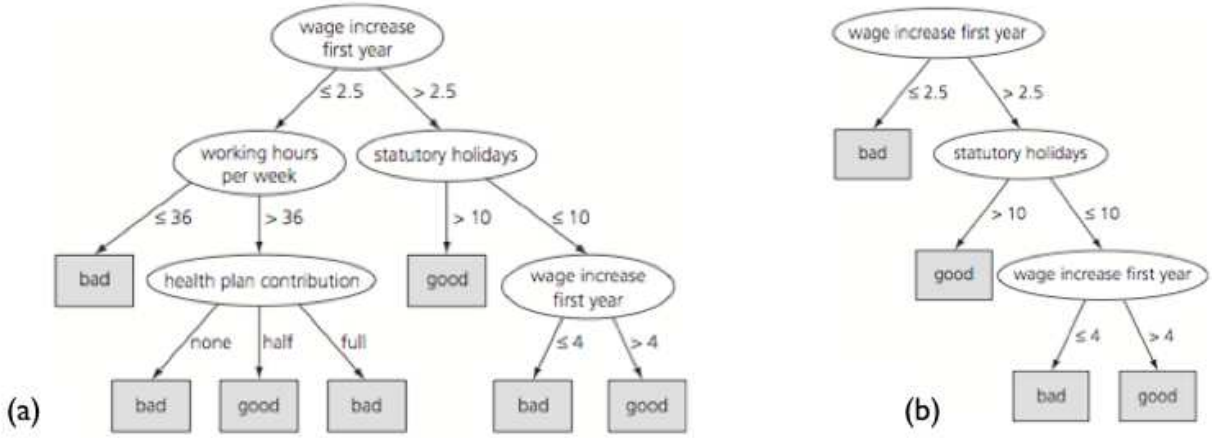

Figure 6 (subtree replacement): *Taken from the book ‘Data Mining: Practical Machine Learning Tools and Techniques’ (modified)*

As we can see, the whole subtree starting at the node *working hours per week* in Figure 6a has been replaced by the its leaf node *bad* in Figure 6b.

2.5.2 Subtree raising

The idea of subtree raising is quite self-explanatory. A subtree that used to be lower down in a tree moves up to occupy a higher position, substituted for what was previously found in that position (Figure 7).

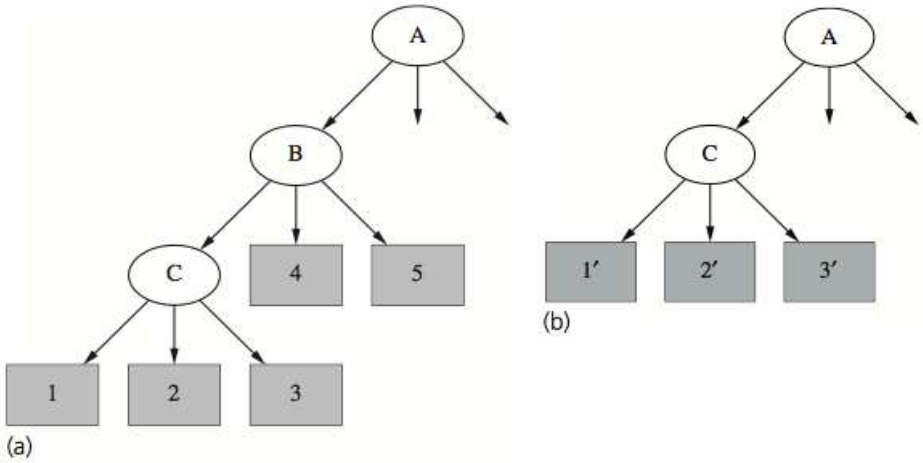

Figure 7 (subtree raising): *Taken from the book 'Data Mining: Practical Machine Learning Tools and Techniques'*

As we see, node C has been raised and substituted for node B.

We have seen in this chapter that there are various ways to build and optimize decision trees. The choice of method is usually driven by the accuracy of classification and a balance must be reached between having a decision tree built based on and optimized for the training data (which therefore classifies those training samples very well) and a tree that is able to perform well on unseen (new) test data. In the next section (section 2.6) we deal with each of the DT classifiers used in our experiments, each one with their own built-in ways of deciding on the optimal final decision tree.

2 .6 DT schemes used in our experiments

For the purposes of classifying our data (OTTO essay collection in English), we have experimented with 10 different decision tree schemes found in the WEKA

package (version 3.6.4): *J48, BFTree, Decision Stump, FT, LADTree, LMT, NBTree, Random Forest, REPTree* and *Simple Cart.* It would be beyond the scope of this thesis to describe each one in detail. Instead, we will briefly comment on 8 of them and discuss 2 of them (J48 and LMT) in more detail. The J48 scheme (an implementation in WEKA of the commonly used C4.5 algorithm) is an algorithm that has a long history in classification and which usually shows very good results. LMT, on the other hand, is a more recently-developed classifier and the one which proved to be the best for our task, not only in terms of classification accuracy but also in terms of better representing the construct we deal with in this thesis, namely, (written) language proficiency.

2.6.1 BFTree

This is a Best First Decision Tree classifier. Instead of deciding beforehand on a fixed way of expanding the nodes (breadth-first or depth-first), BFTree expands whichever node is most promising. In addition, it is able to keep track of the subsets of attributes applied so far and can thus go back and change some previous configuration if necessary. The Gini is the default measurement used for deciding which attribute to split on.

2.6.2 Decision Stump

A Decision Stump is a very simple DT, which is made up of the root node and 3 child nodes (tertiary split). Therefore, a single attribute is selected to split the root node and the 3 created nodes are leaf nodes (at which a classification is made). One of the 3 branches coming out of the root node is reserved for missing values (if any) of the chosen attribute.

2.6.3 FT (Functional Tree)

Instead of checking at a certain point in the tree for one single attribute for all the classes, Functional Trees learn which attributes are more salient for each class at each point (node) in the tree and have the capacity to check for several

attributes at a node, by using a constructor function. This is somehow similar to LMT (however, LMTs tend to be much more compact), which we will shortly discuss.

2.6.4 LADTree

The LADTree scheme (Logitboost Alternating Decision Tree) builds alternating decision trees that are optimized for a two-class problem (the classification problem we deal with in this thesis is a 6-class problem) and that make use of boosting. At each boosting iteration, both split nodes and predictor nodes are added to the tree.

2.6.5 NBTree (Naïve Bayesian Tree)

NBTree is a hybrid classifier: its structure is that of a decision tree as we have seen so far but its leaves are Naïve Bayesian classifiers which take into consideration how probable each feature value (in the training sample) is, given a certain class. In each leaf, the class assigned to a sample is the one that maximizes the probability of the feature values found in this sample. In order to decide whether a certain node should be split or turned into a NB classifier, cross-validation is used.

2.6.6 Random Forest

This algorithm constructs a forest of random trees. Random trees are built by considering at each node a K number of random features (out of F features available) for splitting that node on. This is done for each node and no pruning is performed. The random forest algorithm is a collection of random trees and the class it assigns to a sample item is the mode of the classes assigned to that item by the random trees in the collection.

2.6.7 REPTree

As described in *Data Mining: Practical Machine Learning Tools and Techniques (2nd Edition)*, "REPTree builds a decision or regression tree using information gain/variance reduction and prunes it using reduced-error pruning. Optimized for speed, it only sorts values for numeric attributes once and deals with missing values by splitting instances into pieces, as C4.5 does.".

2.6.8 Simple Cart

Simple Cart is a top-down, depth-first divide-and-conquer algorithm which uses the Gini for deciding which attribute to split on. It uses minimal cost-complexity for pruning and contains classifiers at the leaves.

2.6.9 C4.5 (a.k.a "J48" in Weka)

The C4.5 algorithm was developed by Ross Quinlan (Quinlan, 1993) and builds upon Quinlan's previous ID3 algorithm (Quinlan, 1986). C4.5 is probably the most widely used DT algorithm in machine learning and a benchmark algorithm against whose performance any other algorithm should desirably be compared. It is a top-down, depth-first algorithm and uses a divide-and-conquer strategy. For numerical attributes, C4.5 makes use of binary splits (see figure 8 below) and for nominal attributes (predictor classes) it might use other n-ary splits (binary, tertiary, etc.). The default is to perform post-pruning and in the pre-pruning training process, nodes are split until they are pure (that is, contain only samples belonging to a single class). Information Gain (IG) is used to decide which attribute is used for splitting a certain node and in the post-pruning process estimation of error is calculated by supposing that every sample that reaches a leaf will be classified as belonging to the majority class in that leaf. We can see below in Figure 8 what a typical C4.5 Decision Tree looks like, in this case applied to the weather data set that comes with WEKA:

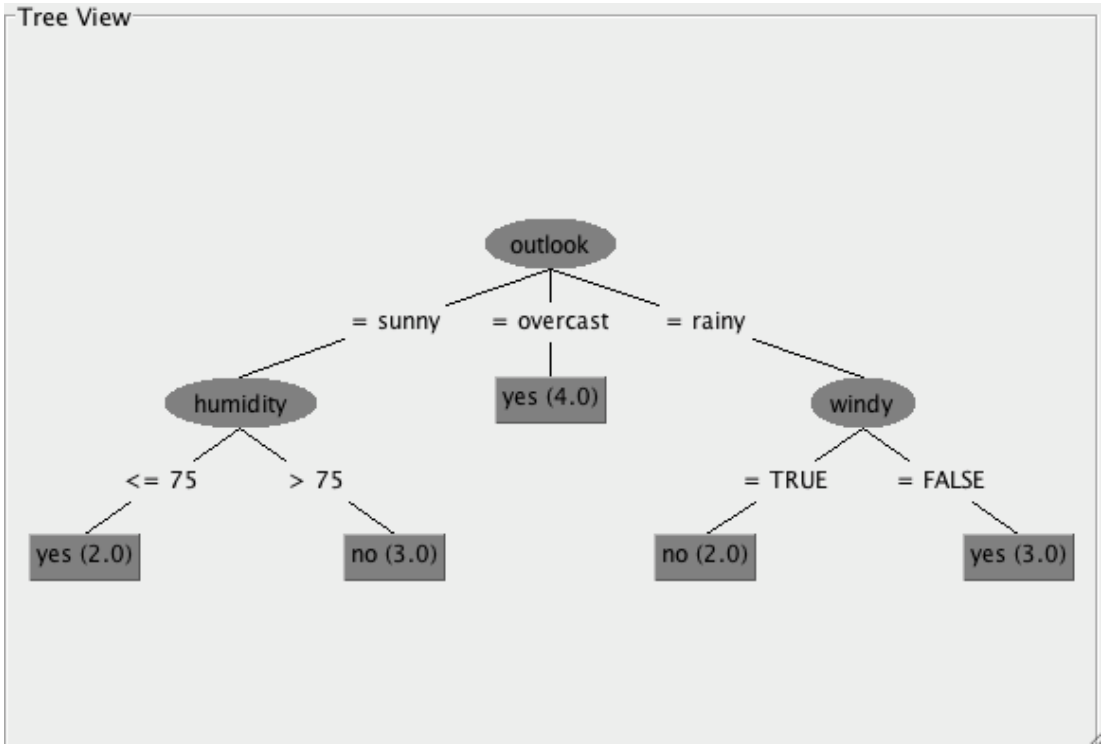

Figure 8: The C4.5 algorithm applied to the *weather* data (visualization taken from WEKA)

2.6.10 LMT (Logistic Model Tree)

A quite recent development in decision tree algorithms is the Logistic Model Tree, or LMT (Landwehr, Hall & Frank, 2005), which has shown quite good results and insights for our particular data and construct and hand (language proficiency level). The algorithm makes use of logistic regression analysis in order to build the tree and, similarly to some of the algorithms seen above, learns not only which independent variables (predictor classes) are most relevant for predicting the dependent variable (target class), but also which attributes (predictor classes) are most relevant to each possible value the target class might take (in our case, levels 0 to 5).The main difference in the approach employed by LMT, however, is that it arrives at a single optimal value of a given attribute for a certain class, thus making the model much more compact than the majority of models above. Therefore, not only is LMT an algorithm that produces more compact trees, but also an algorithm whose results are more intuitive and

easier to interpret. As Landwehr, Hall & Frank put it (2005), "a more natural way to deal with classification tasks is to use a combination of a tree structure and logistic regression models resulting in a single tree" (Landwehr, Hall & Frank, 2005a: 161-205). The authors also note that "typical real world data includes various attributes, only a few of which are actually relevant to the true target concept". We can conclude that LMT seems to be a natural candidate to explain our complex concept/construct: language proficiency.

The basic idea of LMT is to choose from among all the variables in the data, those that are most relevant to each possible value of the target class (these are called *indicator variables)*. By using logistic regression, LMT checks for each possible variable (while holding the others constant) how relevant it is to predicting each of the values of the target variable. The final result of LMT is a single tree, containing multiway splits for nominal attributes (these have to be converted to numeric ones[2], using the usual logit transformation used in logistic regression, in order to be fit for regression analysis), binary splits for numeric attributes and logistic regression models at the leaves, where actual classification is done. At terminal nodes (leaves), logistic regression functions are applied for each possible value (the different levels in our case) of the target class and the relevant indicator variables for that value are checked. Instead of a single predicted class like in the case with standard decision tree schemes, such as C4.5, LMT has at each leaf a logistic regression function for each possible value of the target class, constituting therefore a probabilistic model.

As we can see in Figure 9 below, each indicator value (feature) contains a co-efficient that will be multiplied by the actual value of that feature found in the data sample. Since LMT is an additive model, all the values are added together and whichever class shows the maximum value will be assigned to the data sample. In Figure 9, positive coefficients imply a directly proportional correlation between the indicator variable and the class value at hand and negative ones imply an inversely proportional correlation. During the pruning

[2]For example, instead of using the nominal attributes *hot*, *cold* or *freezing*, we would use temperature ranges instead, such as ^{o}C0 – 12 to represent *cold*.

process, it might even be the case that the tree built will contain only one leaf, making it maximally compact (as is the case with Figure 9 below).

Classifier output

```
Logistic model tree
------------------
: LM_1:35/35 (683)

Number of Leaves  :     1

Size of the Tree :      1
LM_1:
Class 0 :
-31.13 +
[date=october] * 1.44 +
[plant-stand] * -1.34 +
[area-damaged=scattered] * 1.67 +
[leafspot-size=dna] * 1.75 +
[stem-cankers=above-sec-nde] * 8    +
[canker-lesion=dna] * 1.45 +
[fruiting-bodies] * 7.45 +
[external-decay=firm-and-dry] * 3.12 +
[fruit-pods=norm] * 4.41 +
[fruit-spots=dna] * 26.73

Class 1 :
-33.24 +
[int-discolor=black] * 69.21

Class 2 :
-22.61 +
[temp=lt-norm] * 16.96 +
[crop-hist=diff-lst-year] * 2.76 +
[germination=90-100] * -1.58 +
[leaves] * -7.24 +
[stem-cankers=below-soil] * 12.95 +
[canker-lesion=brown] * 9.97 +
[external-decay=firm-and-dry] * 7.87
```

Figure 9: *LMT applied to Weka's soybean data*

Out of the 35 predictor classes present in the soybean data, only a small subset are relevant for the target class in Figure 9: the type of disease that specific soybeans carry (19 possibilities/values for this target class). For one of the possible values of the target class (Class 0 in Figure 9), 10 variables seem to be relevant and for another value (another disease), only 1 variable seems relevant, namely *int-discolor* (Class 1, Figure 9). As we can see, not necessarily the same variables are equally important for all values of the target class.

As Landwehr, Hall & Frank point out (2005), LMT can select relevant attributes in the data in a natural way and the logistic regression models at the leaves of the

tree (one per each value the target class can take) are built by incrementing those present in higher points in the tree. By means of Logitboost (a boosting algorithm), LMT reduces at each iteration step the squared error of the model, but either introducing a new variable/coefficient pair or by changing on of the coefficients in a variable already present in the regression function present at the parent node. What is important to note is that at each iteration step, the training sample available to the model is only those training instances present at that specific node. From the point of view of computational efficiency, it makes more sense to base the logistic regression function at each node on the previous parent node than to start building the model always from scratch.

LMT, just like other DT schemes, must have its own ways of knowing when to stop splitting a node any further and how to prune the tree, once it has stopped growing. In LMT, a node stops being split any further if it meets one of the following conditions:

a) it contains less than 15 examples
b) it does not have at least 2 subsets containing 2 examples each and the split does not meet a certain information gain requirement
c) it does not contain at least 5 examples (this is due to the fact that 5-fold-cross-validation is used by Logitboost in order to decide on the optimal number of iterations it will use).

Once the tree has completely stopped growing, pruning is done by means of the CART pruning algorithm, which uses "a combination of training error and penalty term for model complexity" (Landwehr, Hall & Frank, 2005a:161-205).

As we have seen, each Decision Tree scheme has its own characteristics and ways of deciding on how to classify the samples. We have applied each scheme to our data in order to find out which one seems the most promising for our task of essay scoring. We move on now to describe another approach to classification, namely, a Bayesian one.

3. NAÏVE BAYES

Naïve Bayesian classifiers are simple probabilistic algorithms which apply a slightly modified version of Bayes' Theorem for classification and which make the strong (hence the name *naïve*) assumption that the variables in the data (apart from the target class/variable) are independent from one another. In other words, it assumes that all features F_1 *to* F_n in our data are independent of one another and only the class variable C (in our case, the proficiency level) is dependent on each of the features F_1 to F_n. As Manning and Schütze (1999) put it, citing Mitchell (1997), "Naïve Bayes is widely used in machine learning due to its efficiency and its ability to combine evidence from a large number of features" (p.237). However, as we will shortly see in our language data results, many of the variables are not independent from one another and treating them as if they were might lead to a decrease in the classification accuracy of classifiers such as Naïve Bayes.

A Naïve Bayesian model must first approximate the parameters that will be used by the model in order for it to arrive at a classification. These parameters are the class priors (or class probability) and the feature probability distributions, both of which are calculated based on the training set. A class's prior can be calculated by diving the number of samples in the training set that belong to that class by the total number of samples in the training data (summed over all classes). Thus, the class prior of level 1 in our essay set, for example, would be 131/481, which equals 0.27. The feature probability distributions can be calculated by first separating the data set into the different classes and then calculating, for each attribute in each class, the mean and variance of that attribute in that class. If we take μ_2 to be the mean of the values of X regarding class c, and σ^2_c to be the variance of the values of X regarding class c, then the probability of a certain value of X given a class, $P(x=v \mid c)$ can be found by inserting it in the equation of a normal distribution containing as parameters the mean and covariance of the values of X for a specific class:

$$P(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}$$

In order to make a decision as to which class a certain data sample belongs to, the model calculates the conditional probability of each possible class (in our case, the various English proficiency levels) given the observed values of each of the features present in the data. The Naïve Bayesian probabilistic model is described below:

Probability (C | F_1, F_2, F_3, ..., F_n) = *P (C) * P (F_1|C) * P (F_2 |C) * ... * P (F_n |C) / P ($F_{1...}$ F_n)*

Since the denominator of the formula does not depend on the class and since the feature values are given, we are in practice only interested in the numerator of the right hand side of the equation. Therefore, the probability of a sample belonging to a certain class is given by this updated formula:

$$p(C) \prod_{i=1}^{n} p(F_i|C)$$

We calculate this for each of the possible values of the target class (C) in the data and choose the class whose probability is the highest:

$$\text{classify}(f_1, \ldots, f_n) = \underset{c}{\text{argmax}}\, p(C = c) \prod_{i=1}^{n} p(F_i = f_i|C = c)$$

We have seen that DTs and Naïve Bayesian Classifiers go about the classification task in different ways. In addition, each DT scheme has its own specificities. However, both the DT and Naïve Bayesian approaches try to decide on an optimal classifier configuration based on the features present and their values, so as to increase the accuracy of classification. Depending on the data at hand,

one classifier might have a clear advantage over another and show much better results. It is therefore difficult to tell beforehand which classifier will be better. With this in mind, we have run each of the previously described classifiers on our essay set in order to determine which one is the best for our specific task. We turn to these experiments in chapter 4 below.

4 – PERFORMANCE OF DTs AND NAÏVE BAYESIAN CLASSIFIERS ON OUR LANGUAGE DATA

In order to know which of the classifiers is the best for our task, we must run each of them on our language data and look closely at the results, not only in terms of classification accuracy, but also in terms of the types of misclassification errors, simplicity of classification, adjacent classifications and other factors. In this section, we describe in detail the data we have used in our experiments, the three testing conditions that we have employed and the results of each of the classifiers on our dataset. We also experiment with ways of increasing our accuracy by pre-processing the data and show what the best classifier is for out essay scoring task. Finally, we discuss both the types of misclassifications made by the classifiers as well as possible reasons for those misclassifications.

4.1 Data information

In order to assess the performance of each of the 11 classifiers used in our work (10 DT classifiers and 1 Naïve Bayesian classifier), we have used the 481 essays in the OTTO corpus (see Description of the Data below). We can see in figure 10 below how each of the proficiency levels in represented in the data:

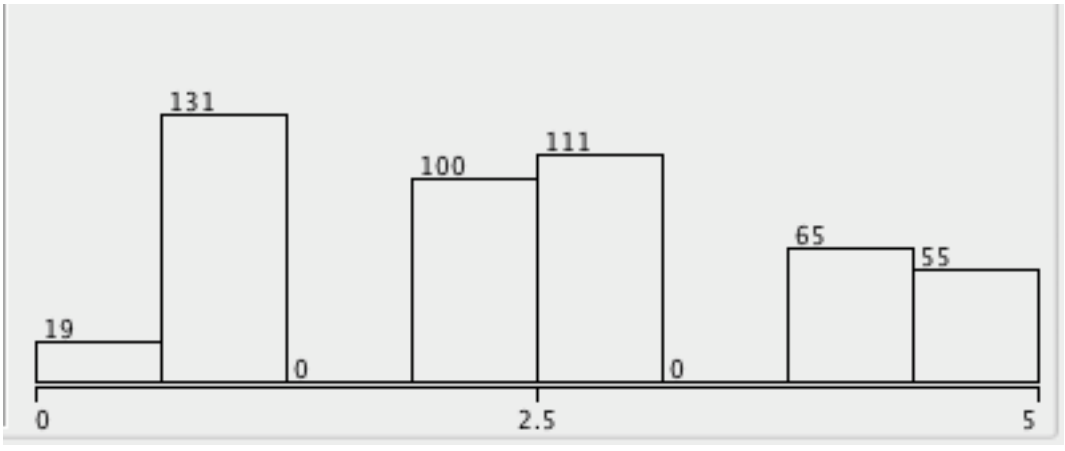

Figure 10 – Distribution of the levels (0 to 5) in our data, as shown in WEKA

All the data used is in an *.xls file* (Excel table), which is converted to a *.csv* (comma separated values) file in Excel itself. The *.csv* file is then converted to an *.arff* file format, which is the native format preferred by the WEKA software.

4.1.1 Description of the data

The corpus was obtained from the OTTO project, which was meant to measure the effect of bilingual education in the Netherlands (www.tweetaligonderwijs).
To control for scholastic aptitude and L1 background, only Dutch students from VWO schools (a high academic Middle School program in the Netherlands) were chosen as subjects. In total, there were 481 students from 6 different WVO schools in their 1st (12 to 13 years old) or 3rd year (14 to 15 years old) of secondary education. To allow for a range of proficiency levels, the students were enrolled in either a regular program with 2 or 3 hours of English instructions per week or in a semi-immersion program with 15 hours of instruction in English per week.

The 1st year students were asked to write about their new school and the 3rd year students were asked to write about their previous vacation. The word limit was approximately 200 words.

The writing samples were assessed on general language proficiency. Human raters gave each essay a holistic proficiency score between 0 and 5. As Burstein & Chodorow (2010) put it, "for holistic scoring, a reader (human or computer) assigns a single numerical score to the quality of writing in an essay" (p.529). In order to ensure a high level of inter-rater reliability, the entire scoring procedure was carefully controlled. There were 8 scorers, all of whom were experienced ESL teachers (with 3 of them being native speakers of English). After long and detailed discussions, followed by tentative scoring of a subset containing 100 essays, assessment criteria were established for the subsequent scoring of essays. Two groups of 4 ESL raters were formed and each essay was scored by

one of the groups. The score of the majority (3 out of 4) was taken to be the final score of the essay. If a majority vote could not be reached and subsequent discussion between the members of that group did not solve the issue, then the members of the other group were consulted in order to settle on the final holistic score for each essay. In all, 481 essays were scored. As we will see further ahead, the size of this set is good enough for training a scoring system and some of the more established Essay Scoring Systems available actually use a smaller set than we do in our work.

The proficiency levels assigned to the essays were calibrated with the writing levels assigned to essays within the Common European Framework (CEF) levels, as can be seen in Figure 11. Level 0, however, does not have a reference in the CEF framework.

Our levels	CEF level
1	Low A1
2	High A1
3	A2
4	Low B1
5	High B1

Figure 11: *Our levels and the CEF framework*

Given that the main interest of Verspoor and Xu was not to assign proficiency levels to the essays but to see how language-learning-related variables might interact and develop within a Dynamic Systems Theory (DST) approach between (and through) the different levels, the authors decided to code as many features (variables) as possible for the annotation of each writing sample, drawn both from the Applied Linguistics literature and from their own observations during the scoring of the essays (Verspoor and Xu, submitted). The features cover several levels of linguistic analysis, such as lexical, structural, mechanical and others. Some of the features used, such as range of vocabulary, sentence length, accuracy (no errors), type-token ratio (TTR), chunks, and amount of dependent clauses, for example, are established features in the literature and used in

several studies to measure the complexity of a written sample. Other features, such as specific types of errors and frequency bands for the word types used in the essay corpus were chosen in order to do a much more fine-grained analysis of language development (for a detailed list of all variables coded for, see the Appendix.) Many of these features are established features in many of the automatic essay scoring systems available.

As mentioned above, in the work by Verspoor and Xu (submitted), which uses the same data as our work here, the annotated features are used with the goal of investigating how these language-related measures develop over time and across levels. In our case, we are interested in using these measurements in order to investigate how they correlate with proficiency level and how they can aid us in our task of automatic essay scoring. Therefore, even though both endeavors use the same data as a starting point, they have quite different objectives.

Description of the features by general areas

The organization of the features used follows (albeit with a few differences) the one used in Verspoor and Xu (submitted) and most definitions and examples are taken from the same article, unless otherwise marked with *NVX*. The description of the features can be found in the Index.
We now proceed to describe the experiments we have conducted. In our first analysis of the classifiers, we decide to keep all 81 features, since all of them might potentially have a strong correlation with proficiency level.

4.2 The three different runs of the experiments

In order to increase the confidence of our estimation as to what the best classifiers are for our task at hand (assessing English proficiency level), we have run 3 different experimental conditions for each of the 11 classifiers:

1) **Super_Test:** we run each classifier through 10 iterations of a stratified

(where class distributions are maintained within each fold) ten-fold cross-validation. This basically means that we run 100 tests on each of the classifiers.

2) **8/9 training, 1/9 test:** For training, we have used stratified 10-fold cross validation on 8/9 of the dataset (non-stratified, random, using weak.core.unsupervised.instances.RemoveFolds). For testing, we have used the 1/9 that was not used in the training phase. Since we have already used stratification for the whole training in the Super_Test above, we have decided to assess as well how each classifier would perform when faced with an even more unpredictable test set.

3) **1 run of 10-cross-fold validation:** In this condition, we do a simple 10-cross fold validation on the data.

We have opted to use 3 different conditions not only to assess the stability of each classifier but also to vary the experimental ways of obtaining our results. What is important is that whenever results are given, they come from the same experimental condition when comparing the performance of different classifiers.

4.3 – Results

In this section, we describe the results of our 11 classifiers on our data.

4.3.1 – Classifier accuracies

The accuracies of the 11 classifiers are shown in Table 1 below. We include here the mean accuracies of each classifier on the Super_Test, the accuracy on the first 5 fold validations in the Super_Test (all in the first iteration still, going from 1,1 to 1,5) and also the accuracy on 8/9 training, 1/9 test. We would also like to draw attention to the fact that the baseline classification accuracy for our data

would be 27%, which is the result of dividing the number of essays belonging to the most common level (level 1 = 131 essays) by the total amount of essays in our corpus (481 essays). We do not include the results of the single 10-cross-fold validation here, but will refer to these later on.

Classifier	Super Test	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	8/9 train, 1/9 test
C4.5 (J48)	50.53	38.77	60.41	50.00	39.58	54.16	57.4
BFTree	49.9	53.06	54.16	50.00	50.00	56.25	50.00
Dec.Stump	40.73	32.65	35.41	43.75	41.66	43.75	33.33
FT	56.07	53.06	56.25	56.25	62,5	62.5	55.5
LADTree	53.49	40.81	52.08	56.25	54.16	56.25	55.5
LMT	58.09	55.10	50.00	66.66	64.58	56.25	64.8
NBTree	45.7	51.02	47.91	45.83	37.5	47.91	51.8
Ran.Forest	53.97	53.06	64.58	66.66	41.66	50.00	46.29
RepTree	51.36	46.93	56.25	64.58	56.25	54.16	53.7
Simple Cart	52.1	55.10	45.83	56.25	50.00	56.25	57.4
Naïve Bayes	52.5	59.18	47.91	58.33	52.08	39.58	55.55

Table 1: *Accuracies (percentage of correct classification) of the 11 different classifiers*

In the table above, the color blue indicates the best accuracy, the color green the second best and red indicates the worst. As we can easily see, there does not seem to be one single classifier which performs the best in every run/test. However, there are two facts we can already notice. Decision Stump is almost always (with one exception) the classifier that performs the worst on the data. It seems however quite impressive that such a simple algorithm (one that uses

only a single attribute for classification) manages to achieve an accuracy as high as 43.75 percent. This is however misleading: the only reason Decision Stump achieves this accuracy is because it classifies every one of the 481 essays into either level 3 or level 1). As we saw in Figure 11 above, these are the two most represented classes in our data. Therefore, this seems like a smart "decision" on the part of Decision Stump and one which will lead to quite a few samples being correctly classified. However, this is not a well informed decision and is not desirable. The Logistic Model Tree (LMT) on the other hand, does seem to qualify as our best classifier so far (we will discuss more details soon), given that in all but one case, it is either the one with the best accuracy or the second best.

4.3.2 The incorrectly classified samples

Looking at classification accuracy is usually enough for deciding on the best classifier to use for a given task. If our task were to classify between different species of animals, for example, then each misclassification is simply wrong: a bear is different from a fish, which is different from a horse, and period. These classes are quite separate and the task at hand is a categorical one. We believe that for a task such as ours, the classification mistakes also matter. Given that our language proficiency classes are ordered, classifying an essay which is in fact level 2 as level 3 is more desirable than the same level 2 essay being classified as a level 5 essay. This holds true for many purposes, be it a placement test at a Language Center or an actual written examination of higher stakes. In addition, scoring agreement between human raters is often not unanimous, which means that a few adjacent classifications might actually be similar to what happens when humans score the essays.

We have therefore developed a system in which we assign a weighted score to each one of our 11 classifiers: 3 points for each correctly classified essay (out of the 481 essays in our data), 1 point for an adjacent classification (level 2 being classified as either 1 or 3, for example) and 0 points for a non-adjacent misclassification. We have decided here to treat an adjacent classification below or above as carrying the same cost for practical purposes. We are nonetheless

aware of the fact that a change in the weights might result in a different classifier ranking. We show in Table 2 below the number of adjacent misclassifications for each of the 11 classifiers in the 8/9 training, 1/9 test condition (54 sample essays are present in the test set) and also the weighted score based on the Super_Test.

Classifier	8/9 train, 1/9 test:*adjacent* vs.*incorrect classifications*	Weighted score on Super_Set (Cor=3, Adj=1, Inc=0)	Weighted- score ranking
LMT	19/19	1013	1
Ran.Forest	24/29	1001	2
FT	23/24	980	3
LADTree	20/24	973	4
Naïve Bayes	19/24	962	5
Simple Cart	19/24	949	6
RepTree	24/25	948	7
BFTree	22/27	908	8
NBTree	21/26	892	9
C4.5 (J48)	17/23	843	10
Dec.Stump	21/36	762	11

Table 2: *Adjacent misclassification and weighted score of all 11 classifiers*

As we can see in Table 2 above, not only are all the misclassifications by LMT adjacent ones, but it is also the classifier that shows the fewest classification errors on the *8/9 training 1/9 test* condition. Moreover, LMT also has the highest weighted score out of all 11 classifiers.

4.4 – The importance of Pre-Processing the data

So far in our experiments, we have used all 81 features and have not subjected our data to any sort of pre-processing. The reasons for not having reduced at first the number of features used for training the classifiers above (which is indeed quite large) were the following:

a) we wanted to assess how each classifier could perform on raw, unprocessed data

b) we want to compare the performance of classifiers when using all features against their performance when using only a few significant features (these features can be found either by doing feature selection at the beginning in WEKA or by running the classifiers and then taking those features shown to be more relevant for classification). We explore the first approach in our work.

c) we wanted to check whether certain classifiers would in some way already do feature selection, that is, use only a subset of the features in their training process (as we have seen, LMT does this in a concise and transparent way).

It is a known fact that obtaining comparable results by using fewer features is a gain in knowledge, given that it makes the model simpler, more elegant and easier to be implemented. Using every feature in order to build a classifier might also be seen as overkill. The question is simple: if we can achieve the same (or possibly even higher) accuracy in a system by using fewer features, why should we use all of them? It takes processing power and engineering/programming work in order for an automatic system to extract the values for each feature and if many of the features do not lead to an improvement in classification accuracy, it does not make much sense to insist on using them if our sole task is classification. In addition, by using too many features we might be missing some

interesting patterns in our data.

By discretizing numerical data (using numerical intervals/ranges instead of a series of continuous values), we are able to build models faster, since numerical values do not have to be sorted over and over again, thus improving performance time of the system. On the other hand, discretizing values leads to a less fine-grained and transparent analysis, since we group together a continuum of values that might have individual significance for classification.

We have experimented with 3 different ways of selecting attributes in WEKA (all of them being classifier independent):

a) *Infogain + Ranker:* The evaluation is performed by calculating the IG of each attribute and the result is a ranking of all features in the dataset, in increasing order of importance.

b) *CfsSubsetEval + Best First:* An optimal subset of features is chosen which correlate the most with the target class ("level", in our case) and the search method is best first (no predefined order)

c) *CfsSubsetEval + Linear Forward Selection:* An optimal subset of features is chosen that correlate the most with the target class and the search method is linear forward selection, a technique used for reducing the number of features and for reducing computational complexity.

All three methods give us quite similar results, in terms of which features seem to be the most relevant. We can see below which features (in increasing order of importance) are selected as being the most indicative of proficiency level in our corpus. We note again that this selection of attributes is classifier independent:

INFOGAIN + RANKER

Figure 12 – *Attribute selection by INFOGAIN + RANKER*

CFS_SUBSET_EVAL + BEST FIRST

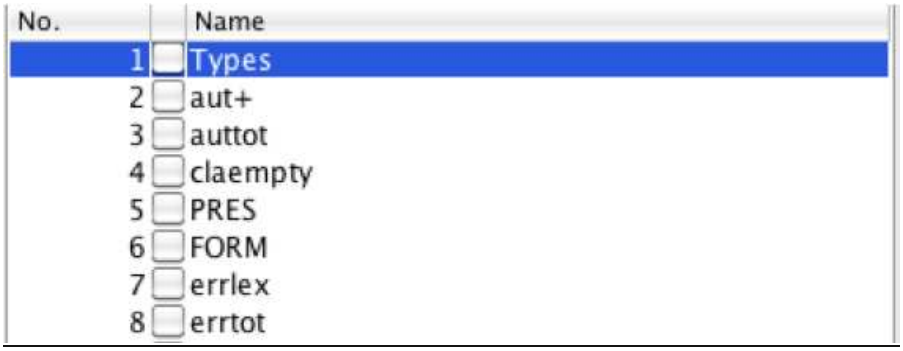

Figure 13 – *Attribute selection by CFS_SUBSET_EVAL + BEST FIRST*

CFS_SUBSET_EVAL + LINEAR FORWARD SELECTION

Figure 14 – *Attribute selection by CFS_SUBSET_EVAL +*LIN.FORW.SELEC.

These 8 features (out of the 81 features present) are the ones that correlate the most (are more indicative of) with proficiency level. Moreover, they suggest that variety, native-sounding structures and errors seem to be the three characteristics of an essay that human beings take the most into account when holistically scoring the essays. As we will see in the next section, using only these 8 features results in an increase in accuracy for our main schemes, given that many noisy or non-relevant features are discarded. A simpler and therefore easier model to be implemented seems to be a better approach to our task.

4.4.1 – New tests with C4.5, LMT and Naïve Bayes

Using only the features available to the classifiers selected by CfsSubsetEval + Best First above (8 features, instead of the 81 or so features previously used), we now present the results of C4.5, LMT and Naïve Bayes on our essay set. We are interested in seeing whether doing feature selection in our task will actually improve the accuracy of our classifiers (besides the obvious advantage of making the search for effective prediction of level easier). As we can see in Table 4 below, we actually manage to improve our classification accuracy by using only these 8 features, which have been found to correlate best with proficiency level. We can therefore conclude that by using all 81 features (many of which do not correlate substantially with proficiency level and can be said to be noisy), the classifiers actually get somewhat confused, so to say, and accuracy is lower. We have used the super-set scheme (10 runs of 10-fold cross validation) in these new tests.

Classifier	Previous accuracy (no pre-processing)	Accuracy (discretization only)	Accuracy (attribute selection only)	Accuracy (attribute selection + discretization)	Accuracy (discr. + attr.sel)
C4.5	50.53%	55.23%	52.93%	58.70%	59.53%
LMT	58.09%	62.29%	60.67%	62.58%	62.27%
Naïve B.	52.50%	60.73%	55.16%	59.09%	60.82%

Table 4: *C4.5, LMT and NB accuracies after pre-processing of data*

As we can see in the table above, either discretizing the numerical values or performing attribute selection has a positive impact on accuracy, when compared to simply using the raw, unprocessed data. The best result, however, seems to come when we perform both attribute selection and discretization in the pre-processing stage. Interestingly, the order in which these two operations are performed affects the performance of the classifiers. By looking at table 4, we can conclude that the best result for both the C4.5 and the Naïve Bayes algorithms comes when discretization is performed before attribute selection. For LMT, however, the accuracy reaches its maximum if discretization is done after attribute selection. Quite surprisingly, in the case of Naïve Bayes, doing only discretization on the data gives us better results than first doing attribute selection and then performing discretization. For all 3 classifiers above, discretization on its own shows more improvement on accuracy than performing attribute selection alone.

We can conclude from the experiments in this section that there is no a-priori best way to pre-process the data. We need to take different classifiers and their respective accuracies into consideration, along with what our task at hand is. If our task is a simple classification one, in which all that matters is classification accuracy, this is what should guide us. However, we should be aware of the fact that discretization leads somehow to loss of more fine-grained information.

We now turn from focusing on accuracy to focusing on the individual contribution of each of the features in our subset to the prediction of proficiency level and to the system as a whole.

4.4.2 Individual contribution of each feature in the subset

We are interested in knowing what the individual contribution of each of our 8 features is to the whole system. Therefore, we have experimented with running LMT in a 10-cross-fold experiment using different conditions. We remind the reader that

our best result so far with LMT was based on the super_set experiment (mean accuracy of 10 runs). Here we use only 1 run of 10-cross-fold iteration, in which accuracy is 64.65% when all 8 features are used. However, the result can be said to be less reliable than in the super_set design. The individual contribution of each feature can be seen below in Table 5:

Feature	Accuracy only using this feature	Accuracy using all other features (7) but this one
TYPES	39.29%	56.34%
AUT+	41.37%	**64.44%**
AUTTOT	44.69%	**62.37%**
CLAEMPTY	37.21%	62.78%
PRES	42.61%	56.75%
FORM	28.48%	62.37%
ERRLEX	34.51%	61.12%
ERRTOT	36.38%	62.16%

Table 5: *Individual contribution of each feature in the subset*

As we can see in the table above, the feature AUTTOT (a sum of both correct and incorrect "native-sounding" structures/constructions) seems to be the feature that correlates the highest with proficiency level when used alone. However, when removed from the subset of 8 features, it does not have as significant an impact on accuracy as the feature TYPES does. We can see, therefore, that our 8 features work as a system and that no feature can be said to be the most important of all. Removing any of our 8 features leads to a decrease in accuracy. Thus, our best option is to use all of them.

In the next section we discuss the misclassification errors that C4.5, LMT and Naïve Bayes have made on our data. We show which errors are more typical (involving which levels) and explore possible reasons for that.

4.5 Misclassification Errors

In this section, we look at what the most typical misclassification error types are for each of the 3 classifiers above (C4.5, LMT and Naïve Bayes). We use the best version of each of these 3 classifiers, namely, the one obtained after performing attribute selection and discretizing the numeric values. Then, we submit our corpus to 1 iteration of ten-fold cross validation in order to analyze the results. Many of the individual essays are misclassified by all three of our classifiers. We discuss these in the next section.

For the moment, we can visualize in Table 6 below the 7 most frequent classification errors by each classifier, along with how many essays were misclassified in that way and how many essays were misclassified in total. The notation 2 = > 3 should be understood as "level 2 gets classified as level 3". Notice that the number of different misclassifications in the table does not add up to the total number of misclassifications, since we only include here the 7 most common misclassification types.

Classifier	Missclas. 1	Missclas. 2	Missclas. 3	Missclas. 4	Missclas. 5	Missclas. 6	Missclas. 7
C4.5	2 => 3 (30/207)	2 => 1 (29/207)	4 => 3 (24/207)	3 => 4 (23/207)	3 => 2 (21/207)	1 => 2 (17/207)	4 => 5 (17/207)
LMT	3 => 2 (24/176)	3 => 4 (20/176)	2 => 3 (20/176)	2 => 1 (20/176)	1 => 2 (19/176)	4 => 3 (18/176)	4 => 5 (14/176)
Naïve Bayes	3 => 4 (23/189)	1 => 2 (23/189)	2 => 1 (22/189)	3 => 2 (22/189)	4 => 5 (18/189)	2 => 3 (16/189)	4 => 3 (15/189)

Table 6 – *Most common misclassification types per classifier*

From the table above we can clearly notice that in the case of all 3 classifiers, the 7 most common classification errors have to do with adjacent classifications, which is exactly what we want for a task such as ours, namely, assigning different proficiency

levels to different students based on their essays. If such a classification system is used in a high-stake scenario, that is, one in which the consequences of the scoring are quite substantial (such as the assessment performed by E-rater in the TOEFL exam, which can define whether a person will be accepted into university of not), an adjacent classification might not be enough[3]. For such situations, nothing short of an extremely accurate classification might be acceptable. However, in other possible scenarios, such as an English placement test within a language center or school, the consequences of an adjacent classification would probably not have such a big impact either on the general system or, psychologically, on the students. Since the classifiers we look at are either accurate or assign adjacent levels in the great majority of cases, it would be simple to move a student a level up or down in the event that some in-classroom discrepancy is noticed. A system such as this, despite not being perfect, would have quite a few advantages, such as making better use of important resources such as teachers' time, not being biased in its classification (increased reliability) and allowing a much bigger number of essays to be analyzed and placements to be done. Other possible uses would be for self-assessment in an online platform and for providing feedback to the student in relation to those features the system takes into account. All this would only be possible, however, once a computational way of extracting these 8 or so features from any essay has actually been implemented and the values can be automatically fed to the classifier. We will discuss this later.

The most common type of misclassification when we look at all 3 classifiers above are: *2 => 1* (71 essays), *3 => 2* (67 essays), *3 => 4* (66 essays) and *2 => 3* (66 essays). These numbers seem to indicate that levels 2 and 3 are the ones that are "tricking" the system the most, so to speak. Even though this might be the case, we cannot affirm this just yet, the reason for that being quite simple. Our levels are not uniformly distributed in the data, as figure 11 (reproduced here as Figure 15) shows.

[3]We note however that in the TOEFL examination, E-rater is used in conjunction with a human rater, which might make an adjacent classification still acceptable for a system. As we will see below, adjacent classifications are also common when only humans are rating the essays.

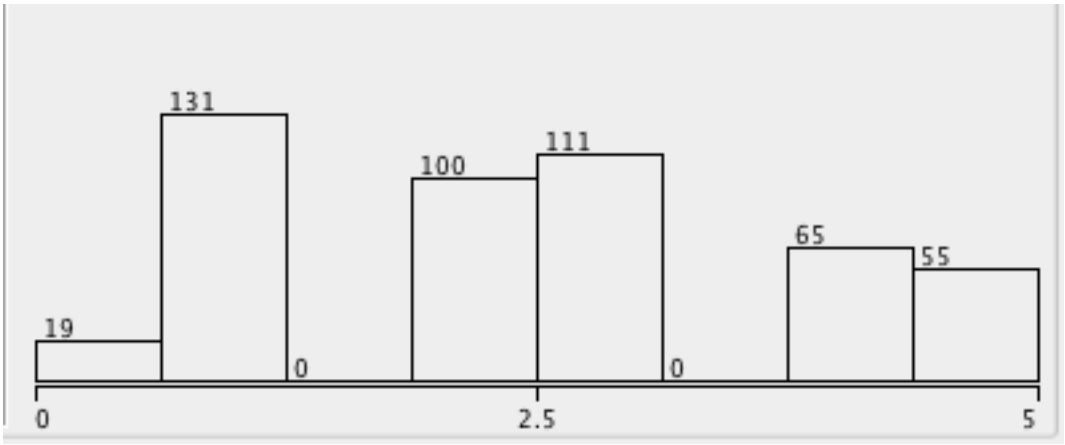

Figure 15 – *Class distribution in the corpus*

Therefore, we must not use absolute numbers, but instead relative numbers, which take class distribution into account. For this, we divide the number of misclassified essays for each level (sum of all 3 classifiers) and divide by the number of essays for that level (multiplied by 3, since we are using 3 classifiers). We can see in Table 7 our updated figures:

Level	Relative Misclassification
0	29 / (19 x 3) = 0.508
1	77 / (131 x 3) = 0.195
2	151 / (100 x 3) = 0.503
3	159 / (111 x 3) = 0.4774
4	110 /(65 x 3) = 0.564
5	46 / (55 x 3) = 0.278

Table 7: *Relative misclassification for C4.5, LMT and Naïve Bayes together*

Our classification errors cannot be said to be only due to the fact that we have a somewhat skewed distribution in our data (some classes are more represented than others). This might apply to levels 0 and 4 somehow, but we see that levels 2 and 3, which have the highest representativeness in the data also get misclassified quite often. Therefore, we cannot say with confidence that the root of the misclassification is lack of enough training data (we will also see ahead that eliminating level 0 from

the corpus does not improve the accuracy significantly). In other words, the reason for misclassification must lie somewhere else and we will try to come up with reasonable hypotheses shortly.

It would be very fortunate if the probability (classification confidence) assigned by the classifiers to all misclassified essays were found to be below a certain threshold and all correctly classified essays above it. If this were the case, we could simply decide not to classify any essays whose probability was below the threshold, preferring instead to trust a human rater with the scoring of those essays. However, this is not the case. Quite often, the classifiers assign misclassified essays a higher classification confidence probability than they do to correctly classified essays.

4.5.1 –Reducing Errors

Given that some of the essays in our corpus have fewer than 25 tokens (which might be too few in order for an automatic system that deals with raw and relative numbers to infer good patterns from data), we decided to experiment with removing these essays from our corpus. The 33 essays that were discarded belong either to level 0 (N=10), level 1 (N=14) or level 2 (N=9). We have run the updated essay collection (448 essays now, instead of 481) again through our best classifier, namely LTM. When no attribute selection or discretization is performed, we manage to increase our accuracy from 58.09% to 59.47% (the super-set scheme was used), which shows that removing those essays might have a positive effect on the system. One of the possible reasons for this (more will be explored later on in the broader discussion of automated essay scoring systems) is that when the system is dealing with raw numbers (which is the case with the TYPES feature), having essays with so few words belonging to a range of 3 different levels (0-2) might confuse the system, since it makes it difficult for the system to find a numerical pattern in the data with regard to this attribute. Surprisingly, if discretization and attribute selection are performed, the effect of removing the essays with fewer than 25 words is actually negative, with precision going down from 62.58% to 61.44%.

We would expect that removing from the corpus both the essays that contain fewer than 25 tokens and also those essays belonging to level 0 (10 out of the 33 essays with

fewer than 25 tokens belong to level 0, a strong correlation) would have a negative effect for the accuracy of LMT, since most of the level 0 essays have fewer than 25 words and the system might use this information accordingly (after all, the TYPES feature is in our selected feature subset). When this is done, the accuracy actually increases from 58.09% to 60.00%. When discretization and attribute selection are applied to the data without the essays with fewer than25 words and with no level 0 essays (TYPES remains in the group of most relevant predictor variables), the accuracy of LMT also decreases on the updated corpus, going from 62.58% to 61.44%. It seems that the advantages of removing these essays from the corpus are lost when discretization and attribute selection are performed. We can conclude that when the attribute TYPES (which tends not to be very different from TOKENS in quite short essays, such as ours) is part of a much smaller set of attributes used in classification, any kind of information available for LMT with regard to feature values is important (specially in the absence of discretization and attribute selection).

Logistic Model Trees are so complex and advanced in their calculation of best predictors for each class and their corresponding coefficients that we might better be guided by a pure accuracy approach when using this classifier. If a certain decision would otherwise make sense (from a testing perspective, for example, it would make sense to exclude essays with fewer than 25 words) but does not increase the system's accuracy (naturally the number of adjacent classifications must be taken into account as well), we should simply not take this specific decision. In the next sections, we discuss the optimal parameters for the classifier most suitable for our essay scoring task: LMT.

4.5.2 Specific Misclassification Errors (by all 3 classifiers, namely, LMT, C4.5 and Naïve Bayes)

In this section, we look more closely at a subset of the essays that got misclassified by all 3 classifiers in the test set-up described in section 4.5 above.

As we will shortly discuss, if we look at LMT's adjacent agreement with human raters, we manage to reach 96% accuracy, which is quite high. On the other hand, an adjacent classification is still a classification error, if we take the human rater's score

to be the definite and correct one. There are quite a few factors that might prevent LMT, C4.5 and Naïve Bayes from correctly classifying a subset of the essays. These are discussed below.

a) *Some essays are simply too short*

As we have seen in section 4.5.1 above, removing from the corpus those essays containing fewer than 25 words leads to an increase in accuracy (when no discretization or attribute selection is performed). The human raters have scored some of those essays as either 0,1 or 2 and for a human, even a little amount of input is enough to judge's someone's language proficiency (think of how easy it is to spot a non-native speaker or how some specific errors simply cannot have been produced by a proficient speaker). For our classifiers, however, which are dealing with either absolute or relative numbers, having too few counts for some features might actually bias the classifiers towards levels in which those feature values are more typical. Human beings are much more difficult to trick in this aspect.

b) The features used are not exhaustive

Even though our 3 classifiers make use of 81 features (many more than the great majority of AES systems do) in the first runs of our tests and 8 features in their updated (optimized) version, there are still some linguistic phenomena which are easily perceived and taken into account by human raters, but which are not recorded in any of the features we use. Let us take one of the essays in our corpus:

> *During our summer holyday we went to Austria. In the beginning it was very nice because we had good weather and there were a lot of nice people to do nice things with. But later on the weather wasn't nice anymore and many people went away. There was also a girl from my age and she also went away. That wasn't nice. But there came some small children and I played with them in the hay. We have seen and done a lot and next year we'll go again to this camping.*

This essay was holistically (taking overall quality into account) scored a level 4 by the human raters and a level 3 by all three classifiers. This essay makes use of some

constructions/structures that show a more refined command of the grammar of the language, such as stranding of prepositions (as in "a lot of nice people to do nice things *with*") and the use of "*there came* some small children[…]". Even though these are constructions that certainly draw the attention of a human rater (since they are more advanced chunks), they only count as another "chunk" in our features and are added to our "AUT+" feature value. There is no distinction between the types of chunks in the AUT+ feature, despite the fact that some chunks are much more typical of advanced students and show a much more fine-grained control of the structure of the language (such as the ones just mentioned). Therefore, including some other features that capture this kind of language use might help towards improving classification accuracy, since these uses are much more typical of proficient than non-proficient language learners.

c) A fundamental difference in the human raters' and the classifiers' scoring procedure

This might be the factor that has the greatest impact on accuracy. The humans raters who scored all 481 essays in our corpus have given great prominence to what can be called "native-sounding" elements in the essays and have consequently scored higher those essays that contained more of these elements. This means, however, that for many raters, punctuation and mechanical errors, for example, did not have much effect on their judgment of the essay's final score, since they do not influence how the essay "sounds". Some of these "native-sounding" structures are captured by our AUT+ feature, which deals with chunks and collocations. Others, such as the ones mentioned in *b* above and the ones in bold below (taken from another essay) are not captured in any special way by any of our features:

> *Hi, my name is Lucca. I'm a* ***freshman*** *at Trevianum. It's way cool here. […] I like doing extreme sports such as: Snowboarding, surfing, Le parkour and riding my dirtbike.* ***Yes, you heard it*** *my dirtbike!*

The essay above was scored a level 5 by the human graders but a level 2 (C4.5) or level 3 (LMT and Naïve Bayes) by the classifiers. The two structures above show knowledge of more refined-vocabulary and of more casual/day-to-day language.

While human raters pick up on these quite effortlessly, this is not fully represented in any of our features (one might say that R5pc, for example, would capture less common words, but it does not make a distinction between them, capturing that some are more "technical" or "casual-sounding" than others). Along the same lines, "you heart it" is simply counted as one more collocation/chunk, despite its quite natural-sounding characteristic. These specific characteristics of words are, however, taken into account by human-raters.

d) *Language itself is a quite complex phenomenon*

Language is a very intricate system, in which all the components (grammar, vocabulary, pronunciation, type of constructions, semantics, etc) interact and develop in often unpredictable ways, as Dynamic Systems Theory shows (Verspoor, de Bot & Lowie, 2004)). Not all students in the same holistic proficiency level show similar feature values for all features. Some use correct spelling, but very simple words. Others, at the same level, may use more complex words that are often misspelled. Some may use correct sentence structure; others may experiment with a more complex sentence pattern and make an error. As Verspoor and Xu show (submitted), there is enormous variation among the learners, especially at the lower levels. However, some of the features, especially aggregated ones, tend to grow (or decrease) linearly across the proficiency levels. Another point is that all subsystems (lexicon, constructions) develop somewhat exponentially (each subsystem becomes more complex) and as the learner becomes more advanced, there are more subsystems that need to develop, making the increments of change at each of these subsystems smaller. The feature subset used in our classifiers (8 features) are all of the more linear type, which explains why using only those 8 features actually improves accuracy, in contrast to using all 81 features. However, there might be other aggregated features that could improve the system further, but are not part of our original feature set, such as bigram or trigram probabilities based on a native corpus, which might capture many of the "native-sounding" structures and uses. Regardless of how advanced a computational system might be, language is still the quintessential area of inquiry where human observers have a clear advantage over automatic systems.

e) A somewhat skewed sample

Many essays in level 0 get misclassified by all 3 classifiers, which might imply that the "calibration" of typical feature values for this level is far from optimal. Given that only 19 out of the 481 essays used for training belong to level 0, we strongly believe that including more essays that belong to level 0 in training would improve the accuracy of the classifiers.

In the automated essay scoring literature, mean scores are often used in order to assess whether the system is on average more strict (classifying essays as a lower level than they actually are) or more lenient, that is, classifying essays as a higher level than actual (Wang & Brown, 2007). Ideally, a system should be neither, but should match the actual classification. However, the implications of either scenario might be worth taking into consideration depending on the use that the system will be put to. It is to the mean scores assigned by LMT that we now turn our attention.

4.6 Mean Scores – LMT (1 iteration of 10 cross-fold validation)

In this section, we explore the mean score assigned by LMT both for the whole scoring task (all levels included) and also on a level basis.

The actual mean score of the whole system is given by the following formula:

Actual mean: (0*19) + (1*131) + (2*100) + (3*111) + (4*65) + (5*55) / 481 = **2.49**

(please refer to Table 8)

The actual mean for each of the levels is simply the actual score at each level. In Table 8 below we can find the actual mean scores and the mean scores calculated from LMT's classification:

Level	Actual Mean Score	LMT's mean score
General (all levels)	2.492	2.494
0	0	0.26
1	1.0	1.15
2	2.0	2.02
3	3.0	3.0
4	4.0	3.87
5	5.0	4.67

Table 8 – *Actual mean scores and LMT's mean scores*

The general mean score assigned by LMT is almost identical to that assigned by the human raters, which means that when taking all levels into consideration, LMT is neither lenient nor strict, performing instead like the human raters. If we look at levels 4 and 5 however, there is a slightly higher discrepancy in the mean scores. As Verspoor and Xu (submitted) found, the more advanced students become, the smaller the differences between adjacent levels. Many of the level 4 essays are actually classified as 3 and many of the level 5 essays as 4. We can also conclude by looking at LMT's mean scores that there is a slight preference for a lower adjacent level than a higher one when it comes to adjacent classifications (which take up the great majority of classification errors). This can be seen in Table 5 above.

4.7 The best classifier and parameters for our task: LMT

After all the different experiments we have conducted in our work, we can clearly say that LMT is the most fitting classifier (out of the eleven classifiers we have experimented with) for our automated essay scoring task. In every single run of the super-set scheme (the most reliable one, given that it performs many more runs and data shuffling than the other schemes used), LMT achieved the best results (see Tables 1, 2 and 4). We can also conclude that the optimal way in which LMT can be used is when we first perform attribute selection followed by discretization during the training phase, leading to an accuracy of 62.58% for LMT. In addition, we should not

remove either level 0 essays or essays with fewer than 25 words from the corpus. If we take adjacent agreement into account, as some results on AES[4] systems do, we manage to achieve an adjacent agreement with human raters of 96%, taking all 5 levels into consideration. The adjacent agreement per level can be found in Table 9 below. Due to a technical issue in WEKA (namely, it does not output a confusion matrix in its Experimenter interface, which is where we run our super-test), our results here are based on a normal 10-cross-fold validation.

	Level 0	Level 1	Level 2	Level 3	Level 4	Level 5
Adjacent agreement	100%	98%	96%	94%	98%	94%

Table 9: *Adjacent agreement for each level (LMT)*

Naturally, the baseline for adjacent agreement is the sequence of 3 consecutive levels that contains the highest number of essay samples. In our case, that would be the sequence of levels 1-3, with respective sample values 131, 100 and 111. By adding all these numbers together and dividing by the total number of essay in the corpus (481), we get the baseline of 71% adjacent agreement.

In Figure 16 below, we include more detailed results per class, as well as the confusion matrix. We note again that this result comes from a 10-cross-fold validation, whereas for Tables 4, 5 and 6 we have used the super-test.

[4]Automatic Essay Scoring

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.586     0.119      0.596     0.586     0.591      0.851    three
                 0.55      0.11       0.567     0.55      0.558      0.837    two
                 0.737     0.009      0.778     0.737     0.757      0.965    zero
                 0.817     0.083      0.787     0.817     0.801      0.94     one
                 0.462     0.077      0.484     0.462     0.472      0.862    four
                 0.727     0.045      0.678     0.727     0.702      0.951    five
Weighted Avg.    0.647     0.089      0.643     0.647     0.645      0.89

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
  65  22   0   2  18   4 |   a = three
  19  55   1  22   2   1 |   b = two
   0   0  14   5   0   0 |   c = zero
   2  19   3 107   0   0 |   d = one
  20   1   0   0  30  14 |   e = four
   3   0   0   0  12  40 |   f = five
```

Figure 16: *More detailed statistics per class (LMT)*

Even though LMT manages to achieve excellent adjacent agreement, there might be several reasons why our accuracy only goes up to 62.58%. These were discussed in section 4.5.2 above.

In sum, the reasons why LMT is the best classifier for our task are several. First, it is a model that manages to drastically reduce the number of features used, making the model not only simpler and computationally efficient, but also leading to a model that has more explanatory power and provides more insights into the problem being dealt with. As Landwehr, Hall & Frank note, “including attributes that are not relevant will make it harder to understand the structure of the domain by looking at the final model, because it is ‘distorted’ by the influence of these attributes” (2005a:167). In addition, LMT is a discriminative classifier, not a generative one. LMT builds through logistic regressions functions a direct mapping between the features input to the logistic regression functions and the class labels. Generative classifiers, on the other hand, must calculate the posterior P (y | x) and then choose the class whose probability is maximal. As we will see in our discussion of how the results of LMT relate to findings in Second Language Development, many of the features available to language learners start showing at different levels. This is in accordance with the feature selection used by LMT, with each class containing in its regression function only those variables which are relevant to that specific class.

4.8 – Pearson's Correlation Coefficient (inter-rater and rater-classifier)

When building an automatic essay scoring system (and many other types of systems), the gold standard, that is, the highest measure possible of performance, is how humans themselves perform the task. With this in mind, we conducted two analyses:

a) Using a set of 25 essays from our corpus that were consistently misclassified by all classifiers, we had a new group of trained raters rate them, in order to check for the correlation coefficient between two groups of human raters.

b) checking the correlation coefficient between the actual scored assigned by the human graders and that assigned by the optimal version of our LMT classifier for all 481 essays in our corpus (1 run of 10-cross-fold validation experiments).

For our analysis, we have used the followed formula for calculation of the correlation coefficient:

Correlation Co-efficient :

$$\text{Correlation}(r) = [\, N\Sigma XY - (\Sigma X)(\Sigma Y) \,/\, \text{Sqrt}([N\Sigma X^2 - (\Sigma X)^2][N\Sigma Y^2 - (\Sigma Y)^2])]$$

where

N = Number of values or elements
X = First Score
Y = Second Score
ΣXY = Sum of the product of first and Second Scores
ΣX = Sum of First Scores
ΣY = Sum of Second Scores
ΣX^2 = Sum of square First Scores
ΣY^2 = Sum of square Second Scores

Figure 17 : *Formula for calculating the correlation coefficient.*[5]

In Table 10 below, we can see the results of the analyses:

[5]http://easycalculation.com/statistics/learn-correlation.php

	Human Raters group 2
Human Raters group 1	0.84
Logistic Model Tree (LMT)	0.87

Table 10: *Correlation coefficients in 2 conditions*

In both cases, we see that the correlation efficient is more than satisfactory. Our LMT classifier performs just as well as a group of humans would. Thus, we can affirm that our classifier is as good as the gold standard.

5. DISCUSSION

In this section, we discuss the relevance and connection of our work in view of the literature on Second Language Development and on Applied Linguistics.

5.1 LMT, our initial features and our feature subset in the context of Automatic Essay Scoring

Automated Essay Scoring has been making substantial progress since its incipience, usually dated to the 1960s and the work of Page and his PEG[6] system (Page, 1966). Many other systems have been developed and others updated since then, such as Intelligent Essay Assessor, ETS1, E-rater, Criterion, IntelliMetric and Betsy, to mention a few. These systems vary considerably in their approaches and methods for essay scoring. In 1996, Page makes a distinction between automated essay scoring systems that focus primarily on content (related to what is actually said) and those focusing primarily on style (surface features, related to how things are said) (as cited in Valenti, Neri & Cucchiarelli, 2003). Intelligent Essay Assessor, ETS1 and E-rater are examples of the former type, while PEG and Betsy[7] (a Bayesian system) are examples of the latter.

The LMT classifier and our approach is more similar to the PEG system developed by Page. Page (1966) defines what he calls *trins* and *proxes*. *Trins* are intrinsic variables such as punctuation, fluency, grammar, vocabulary range, etc. As Page explains, these intrinsic variables cannot, however, be directly measured in an essay and must therefore be approximated by means of other measures, which he calls *proxes*. Fluency, for example, is measured through the prox "number of words" (Page, 1994). In the features used by Dr. Verspoor and Dr. Schmid, the feature TOKENS might be said to be a prox for "fluency" and the feature TTR[8] a prox for vocabulary-richness/range. Both the PEG system and the LMT classifier make use of multiple regression (the former using standard regression and the latter logistic regression). Both types of regression involve calculating the coefficient weights for each feature

[6]ProjectEssay Grade
[7]Bayesian Essay Test Scoring System
[8]Type-Token Ration (Guiraud's index)

and are able to select those features that are most relevant for the classification at hand.

Our feature subset, containing those 8 features/proxes that correlate the most with proficiency level encompass features that are normally used in AES systems. Criterion (an essay scoring and feedback-providing system), for example, analyzes five main types of errors, namely agreement errors, verb formation errors, wrong word use, missing punctuation and typographical errors. All these types of errors are present in our subset of features, in the form of the ERRTOT, ERRLEX and FORM proxes. Many systems use between 30 and even 100 features, whereas ours uses only 8 features and manages to achieve an accuracy of 62.58% (and considerably higher in some runs)in the super-set test and an adjacent accuracy of 98%. The e-rater, for example, extracts more than a hundred features (Kubich, 2000). We must note here that the feature ERRTOT is in fact a bundle of other features that are part of the initial feature set (just as ERRTOT itself is part of the 81 features we start out with). The fact that basically 3 of our 8 final features are related to errors shows just how important error analysis seems to be for an automated essay scoring system and for differentiating between proficiency levels (more on this later).

Two important aspects of our approach to essay scoring (so far) are the following: we only make use of a learner corpus (we have not used any sort of native corpora) and we only analyze the essays for surface features. For our purposes here, which is the automated scoring of essays produced by L2 Dutch younger learners in terms of the level of English proficiency present in the essays, we feel no need to do any sort of content analysis. We are interested in how much control the students have over the grammatical, written and lexical resources of English and thus content (the ways their ideas are expressed in terms of cohesion, coherence and other measures) are not relevant.

5.2 LMT, our initial features and our feature subset in the context of Second Language Development

We analyze here how the features we have used in our study and especially those found to correlate the highest with proficiency level fit with research findings in

Second Language Acquisition (SLA) / Second Language Development (SLD) and also why LMT is the classifier the most fitting for our task.

In the introduction to their 2009 article entitled "Towards an Organic Approach to Investigating CAF in Instructed SLA: The case of Complexity", Norris and Ortega write: "Fundamental to research in several domains of second language acquisition (SLA) are measures that gauge the three traits of complexity, accuracy and fluency (CAF) in the language production of learners" (p.555).

Our initial set of features includes features related to all three of these measures. Examples of complexity measures we have employed are words per utterance (WORDS/UTT), amount of subordination (SYNCPX), amount of present and past tense (PRES and PAST respectively) and others. In relation to accuracy, we have used lexical errors (ERRLEX), amount of incorrect chunks (AUT-), errors in the form of a verb (FORM), errors in the use of a verb (USE), a series of grammatical errors (ERRGRAMs) and several others. Lastly, with regard to fluency, we have looked at the number of tokens in the essay (TOKENS) and also the number of distinct tokens (TYPES), for example.

The subset of 8 features that have shown the greatest correlation with proficiency level in our study have all been reported in the literature on Second Language Acquisition. We move on now to describe how each of the 8 features selected have been shown to correlate highly with proficiency level. We focus especially on the results of the analysis published in Verspoor and Xu (submitted), since they deal with precisely the same dataset and features as we do. However, our analysis is not limited to their study only. Verspoor and Xu (submitted) have decided to exclude level 0 from their analysis, whereas we have decided to keep them.

FEATURE 1: TYPES

As Lu, Thorne & Gamson (submitted) write, "a straightforward measure that has been shown to be potentially useful for measuring child language development is the number of different words (NDW) in a text. Our TYPES feature does precisely that. Even though our feature TYPES has been found to correlate highly with proficiency

level, it does not account for differences in text length. Naturally, a longer text tends to have more types than a shorter one. Some researchers prefer to use Type-Token-Ratio (TTR) or root TTR (Guiraud, 1959), in which instead of dividing the number of types by the number of tokens (normal TTR), the square root of the number of tokens is used in order to account for differences in text length. In our data, TTR has proved not to correlate highly with proficiency level, whereas root TTR is the 3rd feature that correlates the highest. When doing feature selection on the whole set of features, Guiraud's TTR becomes part of the subset. However, despite increasing the accuracy of the system by about 0.8%, it also causes a decrease in the overall precision and recall. For this reason, we have decided to stick to TYPES for our task. In other scenarios, it might be a good idea to use Guiraud's TTR instead of TYPES.

FEATURE 2: AUT+ (chunks/formulaic sequences used correctly)

Doughty and Long (2003) describe ten methodological principles based on SLA[9] research that should be incorporated into any language teaching approach. Encouraging chunk learning is one of these principles, which shows just how important chunks are for language proficiency.

In the study by Verspoor and Xu (submitted), the number of chunks present in an essay has been shown to increase as proficiency level increases, between all levels. This is only natural, given that the more exposure learners have to the target language, the more likely they are to internalize "natural sounding" structures as a single-unit and the more proficient they are likely to become. We can see in Figure 18 below how AUT+ has been shown to develop (in their study, Verspoor and Xu do not make use

of a level 0, however):

[9]Second Language Acquisition

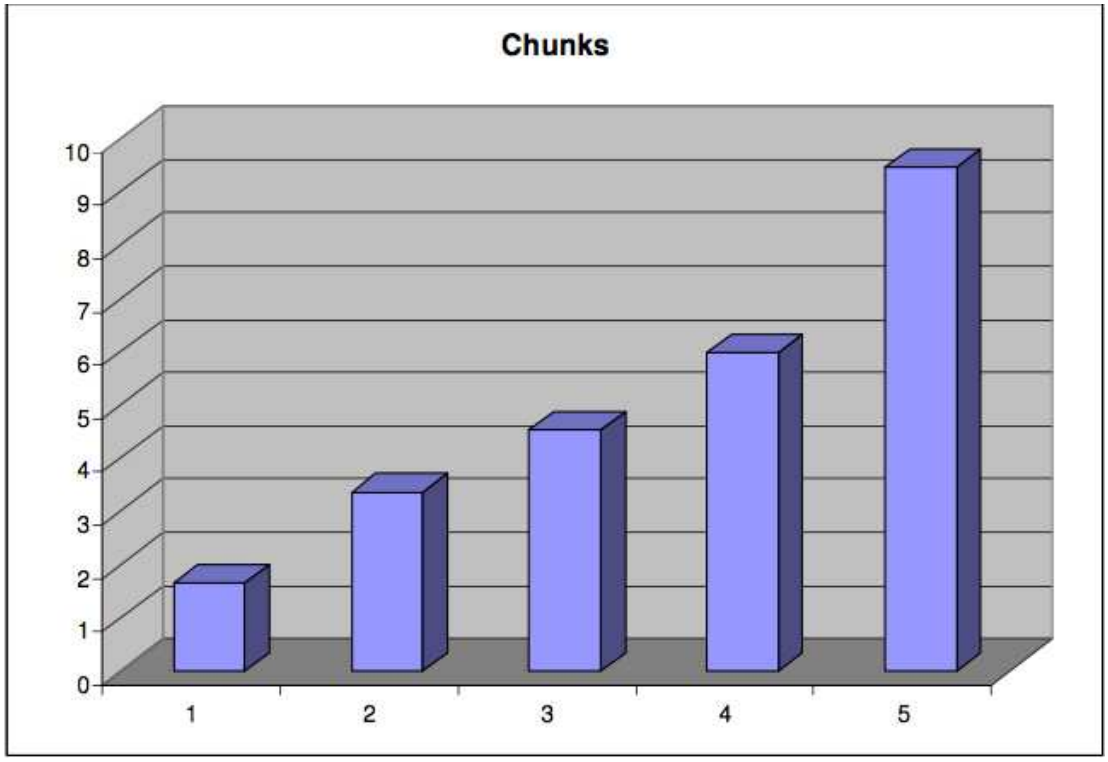

Figure 18: *Development of the AUT+ feature from level 1 to 5. Taken from Verspoor and Xu (submitted)*

FEATURE 3: AUTTOT

Our feature AUTTOT is a combination of AUT+ (correct chunks) and AUT- (incorrect chunks). There are many different kinds of chunks that make up AUTTOT, including collocations, compound words, particles selected by specific verbs/nouns/adjectives along with those verbs/nouns/adjectives. As we have seen, the more a learners uses chunks, the more proficient he seems to be. As Sinclair and Mauranen put it in their work “Linear Unit Grammar: Integrating Speech and Writing” (2006), "The prefabricated chunks are utilized in fluent output, which, as many researchers from different traditions have noted, largely depends on automatic processing of stored units”. According to Erman and Warren's (2000) count, about half of running text is covered by such recurrent units."

On the other hand, using wrong chunks does not necessarily mean that the student is not proficient. There is high variability in the difficulty and transparency of different chunks and the use of wrong ones involves, in the first place, an awareness of the existence of that chunk. Secondly, it shows a willingness to experiment and use newly

learned language. Many of the chunks examined are partial chunks, that is, chunks that have an empty slot and are not fully fixed. The wrong filling of that spot might be responsible for a good percentage of AUT-.

FEATURE 4: CLAEMPTY (clauses without dependent clauses attached)

The more proficient learners become, the fewer simple sentences they will use, giving instead preference to longer and more complex sentences, in which they can tie their ideas in a more coherent way. The amount of subordination has for a long time been used in the SLA literature to represent the syntactic complexity of texts (Ishikawa, 2007, Kawauchi, 2005, Kuiken and Vedder, 2007, Michel et al., 2007). Our feature CLAEMPTY represents exactly the amount of non-subordination/dependent clauses in a text. If the amount of dependent/subordinate clauses has been shown to be quite different between the levels (Figure 19 below), so would the lack of dependent clauses/subordination.

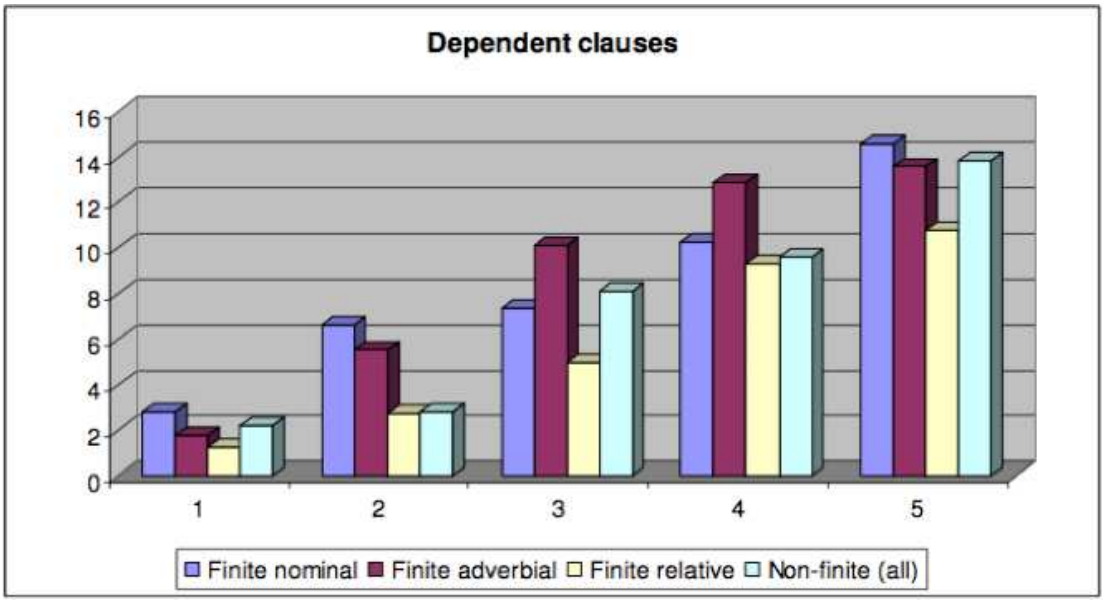

Figure 19: *Development of dependent clauses from level 1 to 5. Taken from Verspoor and Xu (submitted)*

FEATURE 5: PRES (percentage of either Simple Present or Present Perfect)

Our PRES feature revolves around two kinds of verbal constructions: those in the Simple Present and those in the Present Perfect. As we can see in Figure 20below, the more proficient a learner becomes the fewer constructions in the Simple Present they are likely to use, from level 1 to 4. The difference between 4 and 5 is not significant. Conversely, the Present Perfect shows a clear increase from level 1 to level 3 and then decreases from level 3 to 4, showing no real difference between levels 4 and 5 (Figure 21). As we can see, this feature seems to correlate high with the initial proficiency levels and less with the highest levels. In addition, an overuse of Simple Present is probably specific to Dutch as L1, since many sentences which are rendered in English through the Present Perfect, such as *I have lived here for 3 years* are rendered in Dutch in the Simple Present, as in *Ik woon al drie jaar hier*.

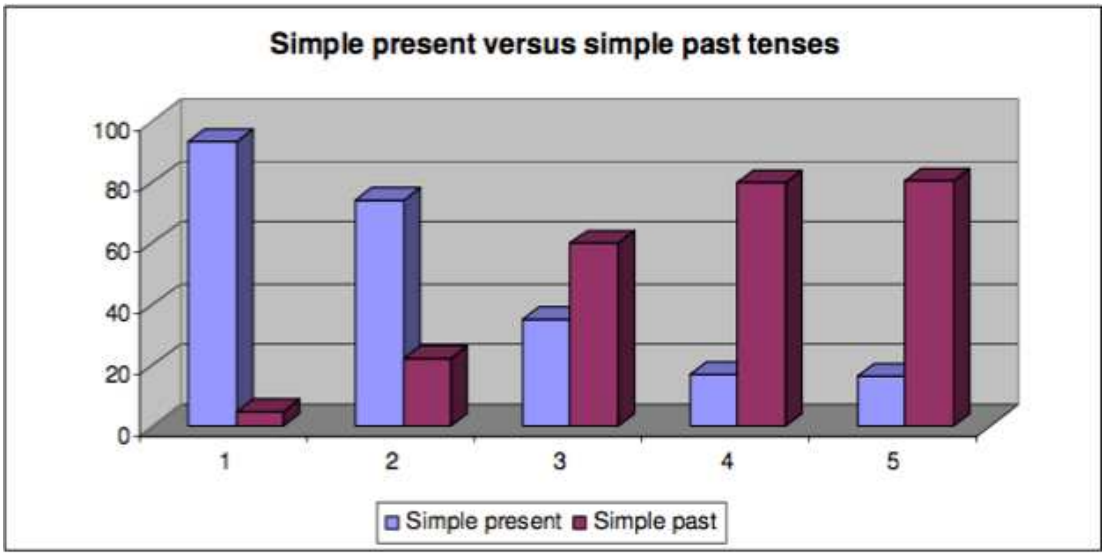

Figure 20: *Development of Simple Present from level 1 to 5. Taken from Verspoor and Xu (submitted)*

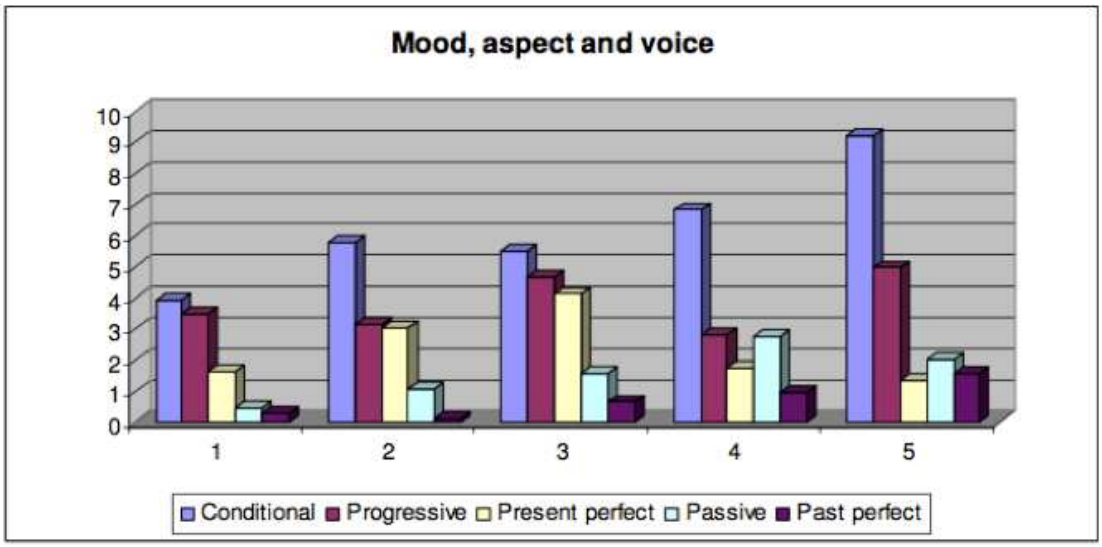

Figure 21: *Development of Present Perfect from level 1 to 5. Taken from Verspoor and Xu (submitted)*

It seems a bit unusual that two features that show an inverse development tendency would be a strong indicator of proficiency level when combined, since we are dealing with a single numerical value here. However, combining different features is quite common in machine learning and if this feature has been selected for our subset, then it is because it is a good idea to combine these two features.

FEATURE 6: FORM (errors in the form of the verb)

The more advanced learners are, the less likely they are to make mistakes related to the form of a verb. It is a known fact that mistakes of the type "He *go* home" or "He *have* seen the movie" are much more likely to be found in the essays of lower level students that in those of higher level ones.

In the paper by Verspoor and Xu (submitted), we can see a clear and linear difference in the number of verb form errors between the different levels (Figure 22). This type of linear difference is exactly the type of feature that has a higher chance of correlating high with the target variable (in our case, proficiency level).

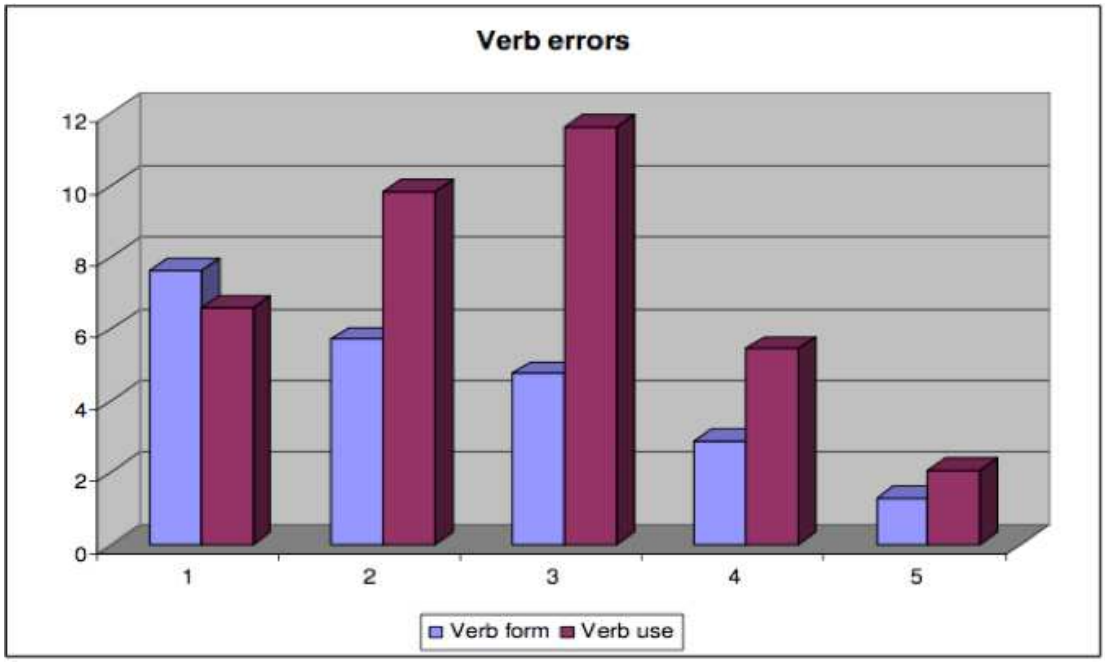

Figure 22: Development in verb form errors from level 1 to 5. *Taken from Verspoor and Xu (submitted)*

FEATURE 7: ERRLEX (lexical errors, summed over all possible subtypes)

With an increase in proficiency in the L2 comes a decrease of the influence of one's L1 on their L2. Therefore, the more advanced students show less L1 (Dutch, in our case) interference on their English. Our ERRLEX feature is in fact the sum of various types of lexical errors, many of which are in fact transfer errors (due to L1 influence). As we can see in the graph below (Figure 23), ERRLEX also shows a clear decrease from level 1 to level 5. The difference between levels 1 and 2, and levels 4 and 5 is ever clearer.

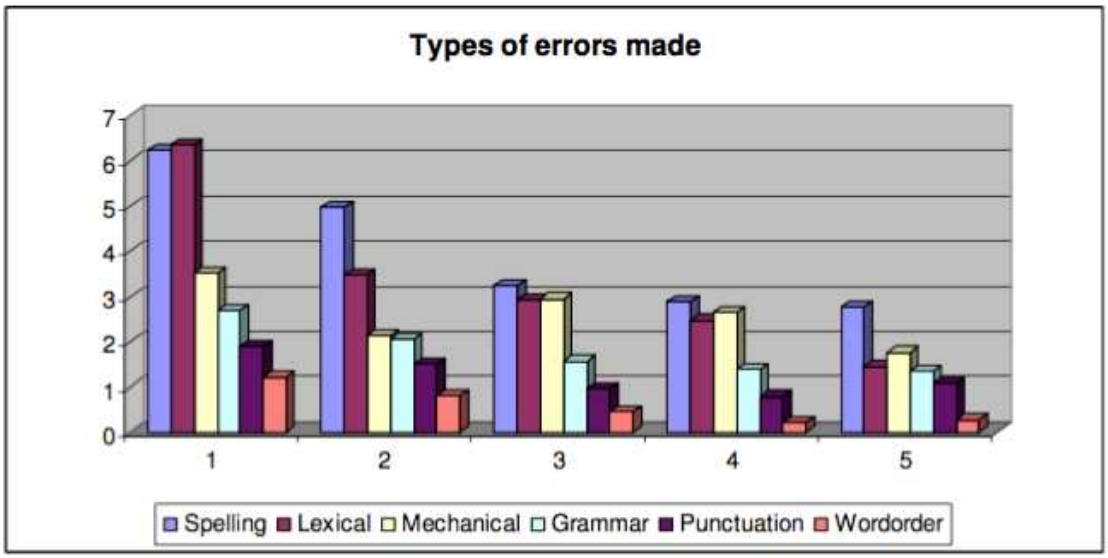

Figure 23: *Development in lexical errors from level 1 to 5. Taken from Verspoor and Xu (submitted)*

FEATURE 8: ERRTOT (total amount of errors)

ERRTOT is a bundle of error types, including lexical, grammatical, punctuation and mechanical. As mentioned in Feature 8 above, the more advanced a student is, the less likely they are to make mistakes, especially more basic ones. Therefore, it is only natural that a feature such as ERRTOT correlates so highly with proficiency level. As speakers of our languages, we can very quickly form an informed idea of someone's proficiency level just based on a kind of mistake they make (and how often). We can see in Figure 24 below how the development of ERRTOT from levels 1 to 5 confirms our statement:

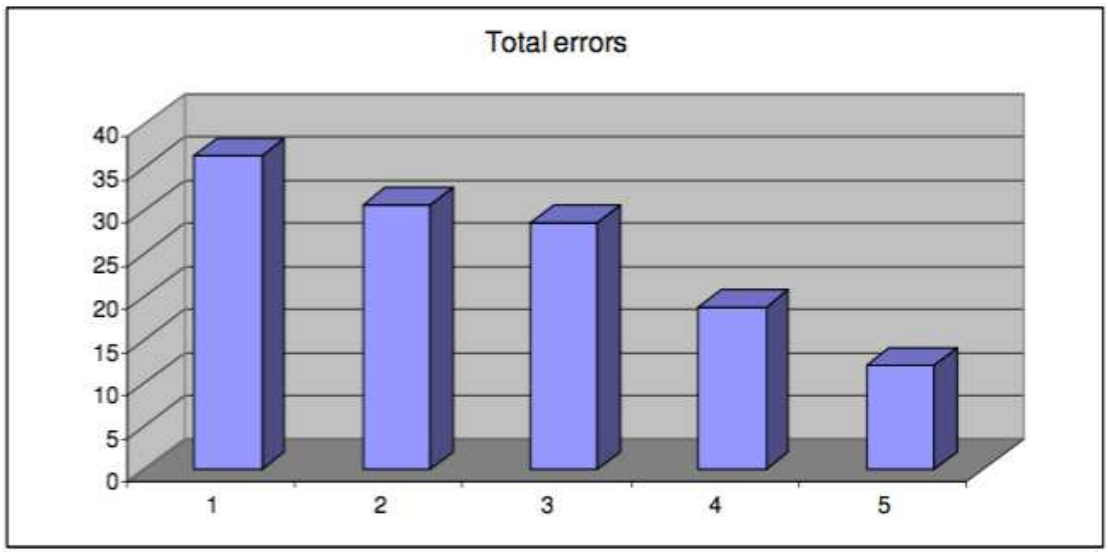

Figure 24: *Development in total amount of errors from level 1 to 5. Taken from Verspoor and Xu (submitted)*

We proceed now to exploring how the values for each of the 8 features in our feature subset might be automatically extracted from an essay.

5.3 – Automation of our 8 features

In this section, we discuss possible ways of automatically extracting the values for our 8 features. As we have seen, LMT performs quite well in terms of classification. However, to have a truly automated essay scoring system, we need to be able to automatically extract the values for each of our 8 features, given a raw essay. These values will subsequently be fed to LMT, which will then output the proficiency level of a specific student. We discuss the automation of the 8 features in the same order in which they are presented in the previous section.

FEATURE 1: TYPES

Out of our 8 features, this is the easiest one to automate. A few lines of code are enough to get the value of TYPES for a given essay. We simply have to count the amount of unique tokens. Some pre-processing is required, however, such as

changing proper names and numerals into a single "NAME or NUMERAL" token. In the former case, we would need to use a subsystem that is able to detect named- entity expressions. In both cases, the use of regular expressions to define the patterns we are after might suffice, since these are essays written by students in either the 1st or 3rd grade and no unusual named-entities or decimal numbers, for example, are likely to be encountered.

FEATURE 2: AUT+ (chunks/formulaic sequences used correctly)

This is arguably one of the most difficult features to automate, not only in our subset, but out of the 81 features we started with. Knowing what constitutes a native-like construction requires an immense amount of training data and exposure to the language, something we humans have probably had in a quantity much higher than any given corpus we might decide to use in an automated system. Our feature AUT+ is actually made up of several types of "native-sounding" structures. Following the examples in Verspoor and Xu (submitted), we show some example of chunks:

a) structures: *better and better , it is easy to do, find it nice, etc.*
b) complements: *decided to, be able to, I don't know what/who/where, etc.*
c) compounds: *sunbathing, deep blue, two-week holiday, etc.*
d) particles: *depend on, go on holiday, make up a story, a group of, etc.*
e) collocations: *the sun goes down, take a dive, hurt badly, etc.*
f) fixed phrases: *lots of fun, have a wonderful time, what a pity, etc.*
g) discourse: *why don't we, in other words, guess what, etc*

We shortly discuss here two main methods that we might employ in order to automatically detect chunks in an essay*: χ^2 (chi-squared)* and *point-wise mutual information.* There are other methods that might be used as well, such as *likelihood interval, likelihood ratio test, Cohran's method* and others. We have decided however to restrict our exploratory discussion to the two aforementioned methods. For both methods we need to have a very large corpus of native English use at our disposal, so as to get our frequency counts (and thus the probability of the constructions). Using only a learner-corpus will not suffice in the case of detecting collocations. In fact, a learner corpus is actually undesirable. We note that automatically detecting chunks is

a quite difficult and complex endeavor and the methods below are more suitable to detect some kinds of chunks than others. Some collocations, particles and fixed phrases for example, can be more easily identified by the methods we will discuss, whereas those chunks that contain partially fixed slots (e.g, *take the bus*) can trick a statistical system much more easily.

a) χ^2 *(chi-squared)*

A chi-squared test works in the following way: it assumes that a *Z* number of variables (words in our case) are independent from each other (this is called the null-hypothesis) and by comparing the observed frequency of co-occurrence with the expected frequency of co-occurrence of these variables, it allows us to conclude whether their observed frequency of co-occurrence is statistically significant. If the answer is positive, we are then forced to reject the null-hypothesis and say that there is a correlation between those variables. The normal experimental design of a *chi-squared* test uses two variables, but it is possible (despite substantially more complicated) to increase the number of variables we input to our chi-squared table. In a 2x2 table, it is important that the number of expected co-occurrences for each cell be at least 5 in order for the chi-squared test to work. We can see in Table 11 below a chi-squared table for calculating whether *take action* might be a chunk:

	ACTION	¬ ACTION
TAKE	A	B
¬ TAKE	C	D

Table 11: *Chi-squared table for calculating whether "take action" is a chunk*

For each cell, we must calculate both the expected and the observed number of co-occurrences. Cell A, for example, represents the expression "take action", whereas cell B represents any expression that begins with the word *take* and is then followed by a word different from *action*. Since many of the words and phrases we might want to check for may not be very common, we need a very large corpus (the web itself is

the most desirable corpus) in order to get reliable counts. We will not go into the details of the calculation chi-squared in here, but we note that in the end, after calculating the necessary numbers, we end up with a single numerical value for that expression we are checking. This final number must be checked against a predefined number in a chi-squared table for the null-hypothesis. If the number output from our table is higher than the number referring to the null hypothesis (different so-called degrees of freedom are possible), then we can say we have a collocation, since our variables co-occur more often than change would grant it.

This method might work quite well for idioms, since there is very little variation in idioms, given that they are a fixed and ordered block of words. Example of idioms are “like a bull in a china shop”, “better later than never”, etc. However, for other kinds of chunks, like “take action”, chi-squared does not work very well, since in the B cell above, we would have quite high numbers as well, given that other chunks starting with *take* such as “take the bus”, “take precautions”, “take office” and “take part” are also common. Another issue is that some chunks might allow a flexible word order, such as “pick the boy up” and “pick up the boy”. Since chi-squared in our case works with a rigid word order, we might miss many counts for certain chunks.

As we can see, even though chi-squared can be quite useful in some cases, it is by no means an exhaustive method for automatically detecting chunks. Point-wise mutual information (discussed below) tends to encounter the same sorts of issues, which might lead us to have to experiment with both statistical and rule-based methods for extracting chunks.

b) point-wise mutual information

Point-wise mutual information quantifies the difference between the probability of the co-occurrence of Z variables given a joint distribution and the probability of their co-occurrence given their individual distributions. The formula for the point-wise mutual information between 2 variables can be found below:

$$\operatorname{pmi}(x, y) = \log \frac{p(x, y)}{p(x)p(y)}$$

The expression “take action” constitutes two variables, the first being the word *take* and the second the word *action*. If we are analyzing, however, if a 4-word expression might be a chunk, however, the formula can be easily adapted (much more simply than chi-squared) to include more variables. In the case of “take action”, we would calculate the PMI between these two words in the following way:

$$\text{PMI (take, action)} = \log \frac{C\text{ (take, action)} / N}{C\text{ (take)} / N * C\text{ (action)} / N}$$

In the formula above **C** stands for the number of times we have seen a specific word (count) and **N** stands for the number of tokens (or words) present in the corpus.
The problem with PMI is similar to the one faced by chi-squared, namely the fact that many chunks are either partial or accept a flexible order. In the former case, we would get a high number in the denominator, since “take” would appear many times in the corpus followed by something else other than “action”. This will lead to a decrease in the probability that “take action” is a chunk. Naturally, we can experiment with different probability thresholds for affirming that a certain expressions is a chunk, but this is not likely to make the system much better.

Given that neither chi-squared nor PMI is enough to automatically detect all types of chunks, a mixture of rule-based and statistical methods might be desirable, with the former taking preference when available. For chunks such as “it is easy to” and “better and better”, a template for these constructions, combined with part-of-speech tagging of both the native corpus and of the essays in question will probably lead to the identification of many chunks which would not be identified by the two statistical methods discussed above. Examples of templates would be *Adjective$_1$ + AND + Adjective$_1$, IT IS + ADJECTIVE + TO* and others.

In sum, the task of automatically detecting chunks in an essay is extremely complex and only a process of trial and error, in which we experiment with many different

techniques such as the one cited above, might lead us towards a system capable of accurately extracting the types of chunks used by Verspoor and Xu (submitted).

FEATURE 3: AUTTOT

As previously mentioned, AUTTOT is a combination of both correctly used chunks (AUT+) and incorrectly used ones (AUT-). AUT- is also complex to be automated. However, the same calculations we have to do for identifying AUT+ might also lead us to extracting AUT-. One possible way to go about the task would be to check for each structure (2 words or more) whether it qualifies as a chunks or not (using chi-squared or PMI, for example). In the case that it is not a chunk, we would check for all the words in our structure, one at a time, if there are other words that could fit in their slot and thus turn the whole structure into a chunk (calculated through the means above). An example would be the structure “like a dog in a china shop”. As we know, this is not a correct chunk, given that the correct chunk would be “like a bull in a china shop”. We would start by calculating the probability that any other X word seen in our corpus in the position of *like* (and therefore before “a dog in a china shop”) might gives us a chunk. The sentence “as a dog in a china shop”, for example, would not qualify as a chunk. However, when we got to the word *dog* and replaced it by “bull”, we would get from of our statistics that the sentence “like a bull in a china shop” does indeed qualify for a chunk. In this way, we can assert that “as a dog in a china shop” is an incorrectly used chunk (AUT-), since there is a slightly different version of it that does qualify as a chunk. This would apply in the same way for incorrect chunks such as “it depends in you” or “I think it nice”, for example. However, it might judge some perfectly fine constructions such as “better and stronger” to be an incorrectly used chunk, since “better and better” might classify as being a chunk. Just as with AUT+, using templates might be a good idea, since something that “almost” fits the template might be judged to be an incorrect chunk. Other incorrect chunks, such as “pick up him” are more difficult to detect. Allowing a flexible word order seems to cause problems for identifying both correct and incorrect chunks.

FEATURE 4: CLAEMPTY (clauses without dependent clauses attached)

Automating our CLAEMPTY[10] feature is somewhat simpler. Dependent clauses are a group of words that do not express a complete thought, despite containing a subject and a verb. Quite often, dependent clauses are preceded by what might be called "dependent marker words". These are words such as *while*, *whether*, *unless*, *when*, *whenever*, *as*, *as if*, *because*, *before*, *even though*, *in order to*, *since*, *though*, etc. If we find one of these words in an essay, there is a good chance that the clause that follows is a dependent clause. The main issue here is identifying the boundaries of the dependent and independent clauses (where each one begins and ends). Such a task can be achieved by means of applying a parser to the sentences. Once the parser identifies a noun phrase (NP) followed by a verb phrase (VP) we know we have a clause. If it follows one of our marker words, then this clause would likely be a dependent clause.

In fact, there are already systems available that are able to identify the number of clauses and dependent clauses in a sentence. One such system is the one developed by Xiaofei Lu (2010), named L2 syntactic Complexity Analyzer. The number of sentences (S), the number of clauses per sentence (C/S) and the number of dependent clauses per clause (DC/C) in an English essay are three of the nine complexity indices that the system is able to identify, by its use of the Stanford parser and a parse-tree querier. With these three numbers, we are able to calculate our CLAEMPTY feature.

FEATURE 5: PRES (percentage of either Simple Present or Present Perfect)

A parser is able to identify syntactic phrases such as noun phrases (NPs), verb phrases (VPs) and others. Many grammar formalisms, such as HPSG and CFG, are able to identify the head of the phrase as well. Once we have identified the head of the VP, we can then analyze it for tense. The present tense in English (both in the Simple and Perfect aspects) is quite easy to analyze, since the only variation is found in the 3rd person singular (such as in *The boy leaves home at 7am*). Therefore, with the help of a parser and a morphemizer (which is capable of identifying specific morphemes in

[10]Percentage of no dependent clauses

words), we are able to get the counts for the feature PRES tense in our essays. Finite-state techniques can also be employed but might not be necessary.

FEATURE 6: FORM (errors in the form of the verb)

Our feature FORM stands for errors in the form of a verb, such as in the sentence *He go to school.* The correct form is *goes*, since the verb must agree with the 3rd person singular subject. Another example of a FORM error would be *He was shoot in the arm.* Grammar formalisms such as HPSG[11] are able to parse complete sentences and, given that it is a unification-based formalism (grammatical features have to match each other incrementally), it identifies problems with agreement, participle forms such as in the passive example above and other types. A formalism such as HPSG would allow us to get our counts for the FORM feature.

FEATURE 7: ERRLEX (lexical errors, summed over all possible subtypes)

The feature ERRLEX is in fact a sum of 7 kinds of lexical errors (cf. Index), including errors caused by L1-Dutch transfer, such as “a long boy”. Tetreault and Charodow (2009), in an article entitled *Examining the Use of Region Web Counts for ESL Error Detection* discuss a new approach to identifying errors in English and an L2/foreign language. By making use of web counts (such as the number of hits a search engine like Google provides), the basic idea is to compare the difference in the frequency of specific constructions (and their variants) in the web counts of a specific non-English speaking region (where English is not a first language) against a region where English is a first language (such as the USA or the UK, for example). In the case of our ERRLEX feature, it might be a good idea to use the Netherlands itself as the only region or one of the non-English speaking regions, since many of the lexical errors in our case are due to transfer from Dutch. A great discrepancy in the number of counts (naturally, different thresholds have to be experimented with) for the non-English speaking regions and the English speaking regions indicates an error. This method circumvents the very common issue of the unavailability of a very large learner corpus (with tagged errors for example) and also avoids the problems

[11] Head-Driven Phrase Structure Grammar

associated with training a model solely on well-formed texts (native essays, for example). However, a combination of this approach with a model trained on tagged learner corpora might prove to be quite useful and complementary.

FEATURE 8: ERRTOT (total amount of errors)

Our last feature in the subset, ERRTOT, is a big bundle of other features, all related to errors. They represent lexical, mechanical, grammar, spelling, mechanics, punctuation, word order and others. The majority of these errors can be identified by the same methods mentioned above, namely, Tetreault and Chodorow's system of using web counts, complemented with a model trained on a learner corpus from Dutch students writing in English. Many of the errors can already be identified by spelling and grammar checkers such as those present in Microsoft Word, for example. Punctuation errors, on the other hand, are likely to be more difficult to be automatically detected, since many parsing models do not take punctuation into account. Another problem with detecting punctuation problems based on web counts is that many of the "hits" provided by Google, for example, come from pages in which people do not pay much attention to punctuation when writing. Therefore, punctuation error detection might be the one type of error that needs to be trained on well-formed corpora. Another possibility for punctuation error detection would be to make use of a Hidden Markov Model of a higher order, such as one implemented through the Viterbi algorithm, trained on a large corpus such as newspaper articles, books, etc. Even here, however, we run into the problem that many of the structures and n-grams used by the Dutch students might not have been seen in the training data, in which case some sort of back-off model would have to be used.

As we have seen in this section, some of the 8 features in the subset lend themselves much more easily to automation than others. AUT+, AUTTOT, ERRLEX and ERRTOT in particular, are much harder to automate. By providing LMT with access to only the 4 features that are the easiest to implement (TYPES, CLAEMPY, PRES and FORM), we manage to keep an accuracy of 55.5%. This is lower than the 62.58% we manage to achieve when all 8 features are used, but shows that once these 4 easier features are implemented in a system, LMT still functions well for our purposes, since the great majority of the misclassifications are still adjacent ones.

6. CONCLUSION AND FUTURE WORK

We have shown in the scope of this thesis that machine learning techniques are quite fitting for the identification of those features that correlate the most with proficiency level. Once we manage to automate the 8 features that correlate the most with proficiency level and extract their values, Logistic Model Tree will prove to be a quite fitting classifier for the task of automatic essay scoring (AES). The LMT scheme/classifier, in particular, not only shows the best results in terms of accuracy and adjacent classifications but also approaches the classification task from a perspective that is more in tune with findings in the Applied Linguistics literature. As Verspoor and Xu (submitted) show, different features develop at a different pace through the levels and not always present a linear behavior. By selecting for each class (proficiency level) only those features that are important for that specific class and calculating the optimal classification coefficient for those features, LMT achieves the best accuracy possible. Moreover, by comparing the correlation coefficients of two groups of humans and that of a group of human versus our LMT system, we conclude that LMT's classification meets the so-called gold standard. In other words, LMT performs just as well for our task and a group of trained human raters would.

We are aware of the fact that we deal here with only part of the proficiency spectrum, since our highest level (level 5) is a high B1 level in the Common European Framework. In addition, we have only used essays written by Dutch students and some of our features might be tuned to phenomena typical of Dutch L1 interference on English, which might lead LMT to perform not so well on essays written by students whose L1 is not Dutch. With regard to the spectrum of our proficiency levels, we have every reason to believe that our system would work just as well if higher proficiency levels were to be included. Regarding the students' L1, only a collection of holistically scored new essays by speakers of different L1s would provide us with the answer as to whether our current classifier would perform well on those essays. In case the accuracy is much lower, all we would need to do is to annotate our 8 features in these new essays and retrain a different classifier. Another possibility would be to merge both classifiers, the one for Dutch and the one for the new L1, so as to create a classifier that would handle more than just one L1.

A logical future step in our work is to develop a system that automatically extracts the values for our subset of 8 features and automatically feeds those to our LMT classifier in order to have a truly automated essay scoring system. Some of the features are certainly easier to be implemented than others, as we have described. In future work, we intend to develop such a system.

7. REFERENCES

Guiraud, P. (1959). *Les charactères statistiques du vocabulaire.* Paris: Presse Universitaires de France.

Ian H. Witten, Eibe Frank, and Mark A. Hall (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3rd edition). Morgan Kaufmann, Burlington, MA.

Ishikawa, T. (2007). The effect of manipulating task complexity along the [+/- Here-and-Now] dimension on L2 written narrative discourse. In M. P. Mayo Gárcia (Ed.). *Investigating Tasks in Formal Language Learning.* Multilingual Matters.

J. Burstein & M. Chodorow. (2010). Progress and New Directions in Technology for Automated Essay Evaluation. In Kaplan, Robert.B (Ed.), *Oxford Handbook of Applied Linguistics* (pp. 529-538). Oxford University Press, 2010.

J. R. Quinlan (1986). Induction of Decision Trees. *Machine Learning* 1:1 , 81-106.

J. R. Quinlan (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

J. M. Sinclair and A. Mauranen (2006). *Linear Unit Grammar: Integrating Speech and Writing.* John Benjamins Publishing Company, Amsterdam.

J.Tetreault and M. Chodorow (2009). Examining the use of region web counts for ESL error detection. *Web as Corpus Workshop* (WAC-5), San Sebastian, Spain.

Kawauchi, C. (2005). The Effects of strategic planning on the oral narratives of learners with low and high intermediate L2 proficiency. In R. Ellis (Ed.), *Planning and Task Performance in a Second Language.* John Benjamins.

Kuiken, F. and I. Vedder (2007). Cognitive task complexity and linguistics performance in French L2 writing. In M. P. García Mayo (Ed.), *Investigating Tasks in Formal Language Learning.* Multilingual Matters.

Kukich, K. (2000)). Beyond Automated Essay Scoring. In M. A. Hearst (Ed.), *The debate on automated essay grading*. IEEE Intelligent systems, 27–31.

Lu, X., Thorne, S. L., & Gamson, D. (submitted). Toward a Framework for Computational Assessment of Linguistic Complexity of Grade-level Reading Materials. Journal of Applied Linguistics.

Lu, Xiaofei (2010). Automatic analysis of syntactic complexity in second language writing. *International Journal of Corpus Linguistics*, 15(4): 474-496

Manning. C and Schutze, H. (1999) *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Michel, M. C., F.Kuiken, & I.Vedder (2007). The influence of complexity in monologic versus dialogic tasks in Dutch L2. *International Review of Applied Linguistics in Language Teaching* 45: 241-59.

Norris, J. M., & Ortega, L.
(2009). *Towards an organic approach to investigating CAF in instructed SLA: The case of complexity. Applied Linguistics*, 30, 555-578.

N. Landwehr, M. Hall, and E. Frank. Logistic model trees. *Machine Learning*, 59(1-2):161–205, 2005.

Page, E. B. (1994). Computer Grading of Student Prose, Using Modern Concepts and Software, *Journal of Experimental Education*, *62*, 127–142.

S. Valenti, F. Neri, A. Cucchiarelli (2003) An Overview of Current Research on Automated Essay Scoring. *Journal of Information Technology Education*, 2, 319-330.

Verspoor, M.H., K. de Bot & W.M. Lowie (2004). Dynamic systems theory and variation: a case study in L2 writing." In H. Aertsen, M. Hannay & R. Lyall, *Words in their places: a Festschrift for J. Lachlan Mackenzie.* Amsterdam: VU, 2004. pp. 407-421

Verspoor, M. and Xu, X. (forthcoming). A dynamic usage based perspective on L2 writing development.

Wang, J. & Brown, M.S. (2007). Automated Essay Scoring Versus Human Scoring: A Comparative Study. *Journal of Technology, Learning, and Assessment, 6*(2). Retrieved June 2011 from http://www.jtla.org.

8. INDEX

DESCRIPTION OF FEATURES USED IN THE STUDY

SENTENCE-LEVEL MEASURES:

Utt: number of utterances in the essay, whereby "utterance" is the same as a T-UNIT, defined by a main clause along with all subordinate clauses attached to it. The sentence "The man called when he got home" is a single utterance, for example.

Words/Utt: average number of words per utterance. This is calculated by dividing the number of words by the number of utterances in the essay. In the single sentence "My teachers are friendly", it is 4.

Synsimp: percentage of simple sentences (containing one finite main clause and maybe including non-finite complex constructions). *Ex: "My teachers are friendly".*

Syncpx: percentage of complex sentences, that is, sentences containing a main clause and at least one finite dependent clause. *Ex:"It was very nice and funny because we buyed all things the same".*

Syncpd: percentage of compound sentences (containing two or more complete main clauses), with "complete" meaning that it is comprised of a subject and a finite predicate. *Ex: "I have very much homework and I have enough to do."*

Syncpdcpx: percentage of compound/complex sentences (with two or more complete main clauses and one or more finite dependent clauses). *Ex: "Now I don't know what to talk about anymore so I just say a lot of things that don't make sense".*

Claadv: percentage of finite adverbial clauses. *Ex: "It was very nice and funny because we buyed all things the same"*

Claempty: percentage of utterances with no dependent clauses. The utterance "I went to Bolivia with my family", for example, has no dependent clauses.

Clanom: percentage of finite nominal clauses (functioning as subject or object). *Ex: "I said I haven't saw them before".*

Clanonfin: percentage of non-finite constructions, functioning as an adverb, nominal or a post-modifier. *Ex: "In de back of the boat were dolphins jumping in our waves".*

Clarel: percentage of finite clauses functioning as a post-modifier of a noun. *Ex: "The most nice thing I've did was mountain biking".*

Synfrag: percentage of incomplete sentences (fragments). ?

Synphras: percentage of incomplete sentences (phrases).Ex: *"A heavy rain".*

VERB-PHRASE MEASURES:

Pres: percentage of verbs that are in the Present (perfect or simple). *Ex: "walks, has gone"*

Pass: percentage or verbs which are in the Passive voice . *Ex: "is written, was written".*

Perf: percentage of verbs in the Perfect aspect (present or past). *Ex: "has gone, had gone".*

Cond: umbrella term for modals, semi-modals, marginal modal verbs and participle verbs used in "if" like constructions. *Ex: "will go, could have gone, went (in: if he went").*

Prog: percentage of verbs in progressive aspect. *Ex: "is walking, was walking."*

CHUNKS:

Aut-: a formulaic sequence not used correctly. *Ex: It goes not with saying that she'll manage.*

Aut+: a formulaic sequence used correctly. *Ex: She remembered it from the top of her head.*

Auttot: sum of *Aut-* and *Aut+* values.

LEXICAL:

Morph: number of morphemes in the essay. The sentence "They left early with the cook-er" has 7 morphemes. NVX

FORM: error in form of a verb*. Ex: "He go to school".*

USE: error in verb use. Ex: *"He has gone to school yesterday".*

Morph/Utt: the ratio between the numbers of morphemes and the number or utterances. Number of morphemes divided by number of utterances.

Tokens: the number of tokens in the essay. The sentence "We arrived there on a Monday" has 6 tokens. NVX

Types: the number of unique tokens in the essay. The sentence "We left because we did not want to stay" has 8 types. NVX

TTR: it is the type/token ratio. In this case, Guiraud's index is used, which is calculated by dividing the number of types by the square root of the number of tokens, so as to avoid a negative correlation with increasing essay length.

R1pc: the percentage of tokens found in a specific essay which are part of the 100-80% bandwidth of frequent tokens used in the whole corpus of essays (that is, the 20% most used tokens within the whole corpus).

R2pc: the percentage of tokens found in a specific essay which are part of the 80-60% bandwidth of frequent tokens used in the whole corpus of essays.

R3pc: the percentage of tokens found in a specific essay which are part of the 60-40% bandwidth of frequent tokens used in the whole corpus of essays.

R4pc: the percentage of tokens found in a specific essay which are part of the 40-20% bandwidth of frequent tokens used in the whole corpus of essays.

R5pc: the percentage of tokens found in a specific essay which are part of the 20-0% bandwidth of frequent tokens used in the whole corpus of essays (that is, the 20% least used tokens).

TypR1pc: the percentage of types found in a specific essay which are part of the 100-80% bandwidth of frequent types used in the whole corpus of essays (that is, the 20% most used types).

TypR2pc: the percentage of types found in a specific essay which are part of the 80-60% bandwidth of frequent types used in the whole corpus of essays.

TypR3pc: the percentage of types found in a specific essay which are part of the 60-40% bandwidth of frequent types used in the whole corpus of essays.

TypR4pc: the percentage of types found in a specific essay which are part of the 40-20% bandwidth of frequent types used in the whole corpus of essays.

TypR5pc: the percentage of types found in a specific essay which are part of the 20-0% bandwidth of frequent types used in the whole corpus of essays (that is, the 20% least used types).

ERRORS:

Errempty: percentage or no error.

Errgram: percentage of grammatical errors (summed over all possible subtypes).

Errlex: percentage of lexical errors (summed over all possible subtypes).

Errmech: percentage of mechanical errors (summed over all possible subtypes).

Errpunct: percentage of punctuation errors (summed over all possible subtypes).

Errspel: percentage of spelling errors (summed over all possible subtypes).

Errwo: percentage of error in word order. *Ex: "I will you pick up".* NVX

Errtot: percentage of errors (summed over all possible subtypes). *".*

Errgram 1: wrong use of apostrophe for plurals or third person singular. *Ex: "weve, do'nt".*

Errgram 2: incorrect use of singular or plural. *Ex: "a very cool teachers".*

Errgram 3: Dutch-like word order involving a verb or a confusion regarding have-be. *Ex: "I have not a friend" or " I like it not".*

Errgram 4: incorrect word form or left-out pronoun. Ex: *"helping very good".*

Errgram 5: a Dutch construction. *Ex: "I have a lot of the lottery", "a shark was escaped".*

Errgram 6: another type of grammatical error.Ex: "how what it was like". NVX.

Errlex 1: use of a Dutch word. Ex: *"wegenwacht, ik, etc".*

Errlex 2: literal translation of a Dutch word into English. *Ex: "A long boy".*

Errlex 3: use of a wrong preposition in a lexical or grammatical chunk due to L1 influence. *Ex: "I'm on this school now".*

Errlex 4: use of a wrong pronoun. Ex: *"It are my best friends".*

Errlex 5: literal translation of a Dutch idiom. *Ex: "I slept with my cousin".*

Errlex 10: adverb/adjective confusion. *Ex: good/well.*

Errlexoth: all other kinds of lexical errors. *Ex: "A (I) like", "the school light (lies)".*

Errmech 1: capitalization error. *Ex: "i".* NVX

Errmech 2: space error. Ex: *schoolstreet.*

Errmech 3: apostrophe error. *Ex: dont.*

Errmech 5: a typo. *Ex: whit.*

Errmech 21: space error not due to transfer from L1 (Dutch). *Ex: ilooked.*

Errmech 22: space error due to transfer from L1 (Dutch). *Ex: olivetree.*

Errmechoth: other mechanical errors.

Errmisvb: percentage of verbs which were missing (but should not be).*Ex: "I want to the school".* NVX

Errpunct: the percentage of errors in punctuation.

Errpunc1: comma splice. *Ex: "I went on holiday with my whole family,we went to a camping and slept in a tent"*.

Errpunc 2: fused sentences. *Ex: "The school is big I like free hours of food".* NVX

Errpunc 3: fragmented sentences. *Ex: "But in the end, when we went back".*

Errpuncoth: other punctuation errors. Ex: *"I have two sisters; Thamires and Thatyana".* NVX

Errspel: total number of spelling errors.

Errsp 1: half-Dutch, half-English words. *Ex: zwimming.*

Errsp 2: phonetically spelled words. *Ex: Franse, to hef.*

Errsp 3: confusing homonyms. *Ex: to/too, see/sea*

Errpel 4: misspelling in difficult words*. Ex: dependent/dependant.*

Errpel 5: other errors. *Ex: heelo/hello*.

Errspel 10: a morphological error. *Ex: easyer*.

Errpel 31: confusing words like *awfull/awful.*

Erroth: errors which are neither lexical, in spelling, in mechanics, in grammar, in word order or in punctuation. *Ex: "I was very happy with my to see my class".*

Errtot: total amount of errors.

NON-LINGUISTIC FEATURES:

TTO: Indicates whether the student attends a bilingual school (around 15 hours a week of English exposure) or a normal school (around 3 hours a week of English exposure).

Grade: student's grade at school (either 1st grade or 3rd grade)

Level: the student's proficiency level as determined by holistic scoring of his/her essay.

END OF INDEX