A review on student automatic grading system D

Diptiben Ghelani[1]

[1]Affiliation not available

September 30, 2022

A review on student automatic grading system

Diptiben Ghelani

Introduction

The majority of papers on automated grading hold keyword matching to be an important factor for evaluating replies. Despite the fact that they are significant, it is normal for people to overlook a few rare words and instead choose synonyms. In this research, a method for automatically grading papers in theory-based topics is presented. This method makes use of natural language processing. Semantic analysis and other machine learning approaches will be used. Matching keywords is ineffective since different students may offer the same response in various ways. Since it is now possible to check for the existence of keywords, synonyms, the proper word combination, and concept coverage, utilising ontology to extract words and their synonyms connected to the domain makes the evaluation process more comprehensive. The aforementioned methods will be put into practise with ontologies and evaluated using common input data made up of technical responses. Following analysis of the data, an automated grading system that is fair, highly accurate, and has a very low mistake rate (compared to a differential human-to-human error rate) will be created for a theory-based topic. The results of a poll of instructors on the criteria they use while manually editing papers were used to develop the algorithm.

Sentiment analysis can be conducted at a word, sentence, or a document level. However, due to the large number of documents, manual handling of sentiments is impractical. Therefore, automatic data processing is needed. Sentiment analysis from the text-based, sentence or document-level corpora is employed using natural language processing (NLP). Most research papers found in the literature published until 2016–2017 employed pure NLP techniques, including lexicon and dictionary-based approaches for sentiment analysis. Few of those papers used conventional machine learning classifiers. Recent years have seen a shift from pure NLP-based approaches to deep learning-based modeling in recognizing and classifying sentiment, and the number of papers published recently on the undertaken topic has increased significantly. The popularity and importance of students' feedback have also increased recently, especially in the

times of the COVID-19 pandemic, when most educational institutions have transcended traditional face-to-face learning to the online mode.

The number of papers published recently indicates a growing interest towards the application of NLP/DL/ML solutions for sentiment analysis in the education domain. However, to the best of our knowledge, in order to establish the state of evidence, the body of literature is lacking a review that systematically classifies and categorizes research and results by showing the frequencies and visual summaries of publications, trends, etc. This gap in the body of literature necessitated a systematic mapping of the use of sentiment analysis to study students' feedback. Thus, this article aims to map how this research field is structured by answering research questions through a step-wise framework to conduct systematic reviews. In particular, we formulated multiple research questions that cover general issues regarding investigated aspects in sentiment analysis, models.

Ohio, Utah, and most US states are using AES systems in school education, like Utah compose tool, Ohio standardized test (an updated version of PEG), evaluating millions of student's responses every year. These systems work for both formative, summative assessments and give feedback to students on the essay. Utah provided basic essay evaluation rubrics (six characteristics of essay writing): Development of ideas, organization, style, word choice, sentence fluency, conventions. Educational Testing Service (ETS) has been conducting significant research on AES for more than a decade and designed an algorithm to evaluate essays on different domains and providing an opportunity for test-takers to improve their writing skills. In addition, they are current research content-based evaluation. The evaluation of essay and short answer scoring should consider the relevance of the content to the prompt, development of ideas, Cohesion, Coherence, and domain knowledge.

Proper assessment of the parameters mentioned above defines the accuracy of the evaluation system. But all these parameters cannot play an equal role in essay scoring and short answer scoring. In a short answer evaluation, domain knowledge is required, like the meaning of "cell" in physics and biology is different. And while evaluating essays, the implementation of ideas with respect to prompt is required. The system should also assess the completeness of the responses and provide feedback.

Several studies examined AES systems, from the initial to the latest AES systems. In which the following studies on AES systems are Blood (2011) provided a literature review from PEG 1984–2010. Which has covered only generalized parts of AES systems like ethical aspects, the performance of the systems. Still, they have not covered the implementation part, and it's not a comparative study and has not discussed the actual challenges of AES systems. Burrows et al. (2015) Reviewed AES systems on six dimensions like dataset, NLP techniques, model building, grading models, evaluation, and effectiveness of the model. They have not covered feature extraction techniques and challenges in features extractions.

This system not covered the comparative analysis of AES systems like feature extraction, model building, and level of relevance, cohesion, and coherence not covered in this review. Ke et al. (2019) provided a state of the art of AES system but covered very few papers and not listed all challenges, and no comparative study of the AES model. On the other hand, Hussein et al. in (2019) studied two categories of AES systems, four papers from handcrafted features for AES systems, and four papers from the neural networks approach, discussed few challenges, and did not cover feature extraction techniques, the performance of AES models in detail. Klebanov et al. (2020). Reviewed 50 years of AES systems, listed and categorized all essential features that need to be extracted from essays. But not provided a comparative analysis of all work and not discussed the challenges.

Readability assessment is an important NLP issue with much application in the domain of language education. The capability to automatically judge the readability of a text would greatly help language teachers and learners, who currently spend a great deal of time skimming through texts looking for a text at an appropriate reading level. Substantial previous work has been done over the past decades(Klare1963,DuBay 2004a, 2004b). Early work generated measures based on simple text statistics by assuming that these reflect the text reading level. For example, Kincaid, Fishburne, and Rodgers (1975) assumed that the lengths of words and sentences represent their respective difficulty. Chall and Dale (1995) used a manually constructed list of words assumed to capture the difficulty of vocabulary. These measures are easy to use but difficult to apply to languages other than English, because some features, such as word length, are specific to alphabetic writing. Such methods, however, do not compete with recent methods based on more sophisticated handling of language statistics. CollinsThompson and Callan (2004)

proposed a classification model by constructing different language models for different school grades (Si and Callan 2001), and Schwarm and Ostendorf (2005) applied a support vector machine (SVM). Both of these methods outperform classical methods and are less language-dependent. These new methods, however, have a serious problem when implementation is attempted for multiple languages: the lack of training corpora. data annotated with 12 school grades have not been at all easy to obtain on a reasonable scale. Another possibility might have been to manually construct such training data, but humans are generally unable to precisely judge the level of a given text among 12 arbitrary levels. The corpora therefore have to be constructed from academic texts used in schools. The amount of such data, however, is limited, and its use is usually strictly limited by copyrights. Thus, it is crucial to devise a new method or approach that allows readability assessment by using only generally available corpora, such as newspapers. Given a single text, it is hard to attribute an absolute readability level from among 12 levels, but given two texts, there should be a better chance of judging which of them more difficult is. This intuition led to the new model presented in this article.

Readability, in general, describes the ease with which a text document can be read and understood. Readability is studied in at least two different domains, those of coherence (Barzilay and Lapata 2008) and language learning. Readability in this article signifies the latter, for both a mother tongue and a second language.

Every method of readability assessment extracts some features from a text and maps the feature space to some readability norm. There are the two viewpoints regarding features and the mapping of feature values to readability, and correspondingly there are two kinds of work in this domain.

Regarding the first type, many researchers have reported how various features affect the readability of text in terms of vocabulary, syntax, and discourse relations. Recently,PitlerandNenkova(2008)presentedanimpressiveverificationoftheeffectsof each kind of feature and found that vocabulary and discourse relations are prominent, although other features are not negligible. Thefocusofthecurrentwork,however,isnotonwhatfeaturesettoconsider,sowe use the same features throughout the article, as explained further in Section 3.1. Rather, the focus of this article is on mapping the extracted feature values to a readability norm. So far, two models have been used for this: regression and classification. In regression, readability is given by a

score based on a linearly weighted sum of feature values. Early methods, from the Wannetka formula (Washburne and Vogel 1928), to the recent methods of Flesch–Kincaid (Kincaid, Fishburne, and Rodgers 1975) and Dale–Chall (Chall and Dale 1995), are of this kind. Elaboration of such regression methods in a more modern context could proceed through a generalized linear model based on estimation of the weights by machine learning, although we have not found such an approach within the literature of readability assessment for language learning. Our proposal is compared with such an enhanced version of regression in Section 8. In classification, readability is segmented by academic grades, and the assessment is conducted as a classification task. The first is implemented by means of statistical classification modeling, as reported in Collins-Thompson and Callan (2004) and Si and Callan (2001). The authors used a language model (unigrams) and a naive Bayes classifier by presuming different language models for each reading level. A language model Mi is constructed for each level of readability i by using different corpora for each level.

The readability of a given text T is assessed using the formula $L(Mi|T) = \Sigma w \in T C(w) \log Pr(w|Mi)$, where w denotes a word in text T, C(w) denotes the frequency of w, and Pr(w|Mi) denotes the probability of w under Mi.

The second is based on an SVM (Schwarm and Ostendorf 2005) and the authors also studied the effect of statistical features, such as n-grams and syntactic features. In these papers, the readability norms are represented by means of scores and classes of readability. That is, given a single text, the system assigns a value corresponding to a school grade. The result is easy to understand, and various applications have been constructed with this type of scoring. This solution only works, however, when a sufficient amount of training data with annotations regarding multiple levels is provided. Usually, the availability of training data in readability assessment is limited, even for school grading. This is due to the inherent difficulty of classifying the readability of a text into12 grades, making it difficult to uniformly construct a large set of training data. Moreover, the copyright issue is more serious for academic texts. Given this situation, when readability assessment is modeled by regression or classification, a research team wanting to apply these previous methods faces the problem of assembling training data, as we did for over a year.

The readability norm is designed in a completely different way: Given two texts, a comparator judges which more difficult is. By applying this comparator, a set of texts is sorted. The readability of a text is assessed by searching for its position.

Gaining new knowledge, in both formal and informal environments, relies heavily on learning from text. An important component of the comprehension process is the difficulty of the text being read. Texts that are too difficult can impede comprehension. Educators find texts that are grade appropriate and may also need to select texts that meet the need of individual student. Given the abundance of text materials available, educators simply do not have the time to find and thoroughly evaluate texts for this purpose. As such, educators depend on text difficulty formulas to quickly identify appropriately challenging texts. Common readability formulas such as Flesch-Kincaid Reading Ease (Flesch, 1948) that assess text difficulty are usually based on number of syllables per word, number of words per sentence, and the number of sentences (Klare, 1974). Though easy to use, these formulas are centered on relatively shallow lexical and sentential indices. However, theories of reading comprehension suggest that deep features related to syntax and semantics drive text difficulty (Dufty et al., 2006; Duran et al., 2007; McNamara, Graesser, and Louwerse, 2012). To address this issue, researchers have developed natural language processing (NLP) tools that extract richer information about the linguistic features of a text that reflect complex dimensions such as narrativity, syntactic complexity, and cohesion (e.g. Crossley et al., 2016; McNamara, et al., 2014). Researchers have also begun to employ machinelearning approaches for measuring text readability (Collins-Thompson, 2014; Kate et al., 2010; Kotani, Yoshimi, and Isahara, 2011; Pilán, Volodina, and Johansson, 2014). These approaches have shown promise in more accurately assessing text difficulty as compared to "classic" readability approaches (François and Miltsakaki, 2012). Though promising, most of this work has focused on either determining the best set of linguistic features or comparing regression and classification approaches (e.g., François and Miltsakaki, 2012; Heilman et al., 2008). To our knowledge, there is little work investigating the potential for hierarchical approaches in the classification of text difficulty. Hierarchical classification has been used in a number of areas such as protein classification (Zimek et al., 2008), essays scoring (McNamara et al., 2015), and automatic target recognition (Casasent and Wang, 2005). This study addresses this gap in the literature by combining NLP and machine learning to compare multiple types of classification in their accuracy of classifying text difficulty. We first provide brief description of

the relevant NLP tools and machine learning techniques and then present results of the experiments.

LP intersects computational linguistics, computer science, and artificial intelligence to understand, assess, and respond to naturally occurring human language. NLP has been used in education to support student learning, for intelligent and automatic assessments, to improve learning and teaching in massive open online courses (MOOCs), and to develop learning systems. In this study, we employed the NLP tool, Coh-Metrix (McNamara et al., 2014), which integrates a number of sophisticated tools such as advanced syntactic parsers, part-of-speech taggers, distributional models, and psycholinguistic databases (Coltheart, 1981) to generate over 400 indices of language, text, and readability.

Machine learning algorithms are categorized as unsupervised and supervised. Unsupervised learning uses data that is not labeled, whereas in supervised machine learning, the algorithms are trained on labeled data. For supervised algorithms, regression is used to predict quantitative variables, whereas classification is used to predict qualitative variables (Hastie, Tibshirani, and Friedman, 2009; James et al., 2013). As our data involves human ratings of categories (i.e. labeled categorical data), we adopted a supervised learning classification approach. Commonly used classification algorithms include Decision Trees, Naïve Bayes, Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), Logistic Regression, Random Forests, Neural Networks, and Boosting (for further description of these algorithms, see Balyan, McCarthy, and McNamara, 2017; Hastie et al., 2009). Preliminary experiments with a number of these algorithms indicated that SVM and LDA were the most accurate for the current data.

Most natural language processing (NLP) problems can be formulated as classification problems (given some object and its context, decide on the class of this object). Typical instances of this type of problem are part-of-speech tagging and word sense disambiguation. Supervised learning methods work by extracting regularities from a set of examples (e.g. collected from an annotated corpus). The reason why these methods are researched intensively is that, like statistical approaches, they are often reported to achieve higher efficiency, more robustness, and better coverage than handcrafting approaches. On top of this, they are reported to have a number of advantages compared to statistical approaches. E.g., ILP systems allow easy integration of

linguistic background knowledge in the learning system, induced rule systems are interpretable, memory-based learning methods incorporate smoothing by similarity-based learning, etc.

Machine learning is a multi-disciplinary field having a wide-range of research domains reinforcing its existence. These are as shown in the following figure 1. The simulation of ML models is significantly related to Computational Statistics whose main aim is to focus on making predictions via computers. It is also co-related to Mathematical Optimization which relates models, applications and frameworks to the field of statistics. Real world problems have high complexity which make them excellent candidates for application of ML. Machine learning can be applied to various areas of computing to design and program explicit algorithms with high performance output, for example, email spam filtering, fraud detection on social network, online stock trading, face & shape detection, medical diagnosis, traffic prediction, character recognition and product recommendation amongst others. The self-driving Google cars, Netflix showcasing the movies and shows a person might like, online recommendation engines—like friend suggestions on Facebook, "more items to consider" and "get yourself a little something" on Amazon, and credit card fraud detection, are all real-world examples of application of machine learning.

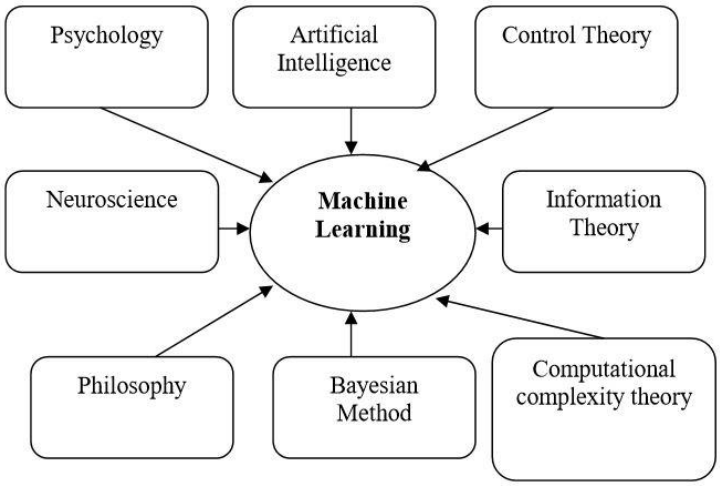

Figure: The multidisciplinary machine learning

Data Science problems and Machine Learning

Machine learning is required to make the computers sophisticatedly perform the task without any intervention of human beings on the basis of learning and constantly increasing experience to understand the problem complexity and need for adaptability.

Machine learning has proven capabilities to inherently solve the problems of data science. Hayashi and Chikio [3] define data science as, “a concept to unify statistics, data analysis, machine learning and their related methods in order to understand and analyze actual phenomena" with data”. Before taking to problem solving, the problem must be categorized suitably so that the most appropriate machine learning algorithm can be applied to it.

Thus depending on the type of problem, an appropriate machine learning approach can be applied. The various categories are explained below:

Classification Problem

A problem in which the output can be only one of a fixed number of output classes known apriori like Yes/No, True/False, is called a classification problem. Depending on the number of output classes, the problem can be a binary or multi-class classification problem.

Anomaly Detection Problem

Problems which analyze a certain pattern and detect changes or anomalies in the pattern fall under this category. For example, credit card companies use anomaly detection algorithms to find deviation from the usual transaction behavior of their client and raises alerts whenever there is an unusual transaction. Such problems deal with finding out the outliers.

Regression Problem

Regression algorithms are used to deal with problems with continuous and numeric output. These are usually used for problems that deal with questions like, ‘how much’ or ‘how many’.

Clustering Problem

Clustering falls under the category of unsupervised learning algorithms. These algorithms try to learn structures within the data and attempt to make clusters based on the similarity in the

structure of data. The different classes or clusters are then labeled. The algorithm, when trained, puts new unseen data in one of the clusters.

Reinforcement Problem

Reinforcement algorithms are used when a decision is to be made based on past experiences of learning. The machine agent learns the behaviour using trial and error sort of interaction with the continuously changing environment. It provides a way to program agents using the concept of rewards and penalties without specifying how the task is to be accomplished. Game playing programs and programs for temperature control are some popular examples using reinforcement learning.

Development of Machine Learning

The words, Artificial Intelligence and Machine Learning are not new. They have been researched, utilized, applied and re-invented by computer scientists, engineers, researchers, students and industry professionals for more than 60 years. The mathematical foundation of machine learning lies in algebra, statistics, and probability. Serious development of Machine Learning and Artificial Intelligence began in 1950's and 1960's with the contributions of researchers like Alan Turing, John McCarthy, Arthur Samuels, Alan Newell and Frank Rosenblatt. Samuel proposed the first working machine learning model on Optimizing Checkers Program. Rosenblatt created Perceptron, a popular machine learning algorithm based on biological neurons which laid the foundation of Artificial Neural Network.

1950	Alan Turning created "Turning Test" to check a machine's intelligence. In order to pass the Turning Test, the machine should be able to convince humans that there they are actually talking to a human and not a machine.
1952	Samuel created a highly capable learning algorithm than can play the game of Checkers with itself and get self-trained.
1956	Martin Minsky and John McCarty with Claude Shannon and Nathan Rochester organized a

	conference in Dartmouth in 1956 where actually Artificial Intelligence was born.
1958	Frank Rosenblatt created Perceptron, which laid the foundation stone for the development of Artificial Neural Network (ANN).
1967	The Nearest Neighbor Algorithm was proposed which could be used for “Pattern Recognition”.
1979	Stanford University students developed “Stanford Cart”, a sophisticated robot that could navigate around a room and avoid obstacles in its path.
1981	Explanation Based Learning (EBL) was proposed by Gerald Dejong, whereby, a computer can analyze the training data and create rules for discarding useless data [7]
1985	NetTalk was invented by Terry Sejnowski, [8] which learnt to pronounce English words in the same manner that children learn.
1990s	The focus of Machine Learning shifted from Knowledge-driven to Data Driven. Machine Learning was implemented to analyze large chunks of data and derive conclusions from it [9]
1997:	IBM invented the Deep Blue computer which was able to beat World Chess Champion Gary Kasparov.
2006	The term “Deep Learning” was coined by Geoffery Hinton which referred to a new architecture of neural networks that used multiple layers of neurons for learning.
2011	IBM’s Watson, built to answer questions posed in a natural language, defeats a Human Competitor at Jeopardy Game.
2012	Jeff Dean from Google, developed GoogleBrain, which isa Deep Neural Network to detect patterns
2014	Facebook invented the “DeepFace” algorithm based on Deep Neural Networks capable of
2015	Toyota Invests $1 Billion in Artificial Intelligence in U.S.
2016	The Year That Deep Learning Took Over the Internet
2017	Facebook & Microsoft Joined Forces To Enable AI Framework Interoperability

2019	AI set to boost cybersecurity industry
2020	AI hardware continued to develop in 2020, with the launch of several AI chips customized for specialized tasks.
2021	The NVIDIA DGX A100 is the first computer of its kind in New Zealand and is the world's most advanced system for powering universal AI workloads
2022	the release of GPT-3 by OpenAI, the most advanced (and largest) language model ever created,

M4

The generic model of machine learning consists of six components independent of the algorithm adopted. The following figure depicts these primary components.

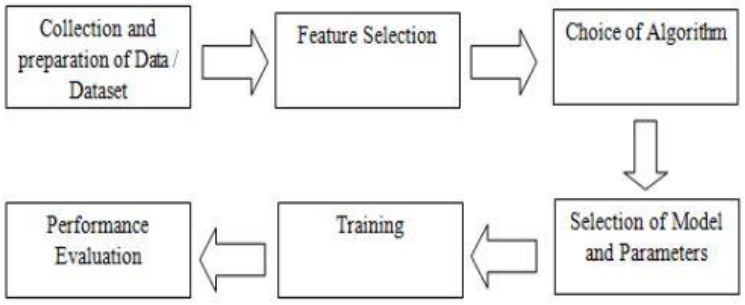

Figure: Components of a Generic ML model

Each component of the model has a specific task to accomplish as described next.

i. Collection and Preparation of Data

The primary task of in the machine learning process is to collect and prepare data in a format that can be given as input to the algorithm. A large amount may be available for any problem. Web data is usually unstructured and contains a lot of noise, i.e., irrelevant data as well as redundant data. Hence the data needs to be cleaned and pre-processed to a structured format.

ii. Feature Selection

The data obtained from the above step may contain numerous features, not all of which would be relevant to the learning process. These features need to be removed and a subset of the most important features needs to be obtained.

iii. Choice of Algorithm:

Not all machine learning algorithms are meant for all problems. Certain algorithms are more suited to a particular class problem as explained in the previous section. Selecting the best machine learning algorithm for the problem at hand is imperative in getting the best possible results.

iv. Selection of Models and Parameters:

Most of machine learning algorithms require some initial manual intervention for setting the most appropriate values of various parameters.

v. Training:

After selecting the appropriate algorithm and suitable parameter values, the model needs to be trained using a part of the dataset as training data.

vi. Performance Evaluation:

Before real-time implementation of the system, the model must be tested against unseen data to evaluate how much has been learnt using various performance parameters like accuracy, precision and recall.

Machine Learning Paradigms Depending on how an algorithm is being trained and on the basis of availability of the output while training, machine learning paradigms can be classified into ten categories. These include: supervised learning, semi-supervised learning, unsupervised learning, reinforcement learning, evolutionary learning, ensemble learning, artificial neural network, Instance-based learning, dimensionality reduction algorithms and hybrid learning. Each of these paradigms is explained in the following sub-sections.

Supervised Learning

Under supervised learning, a set of examples or training modules are provided with the correct outputs and on the basis of these training sets, the algorithm learns to respond more accurately by comparing its output with those that are given as input. Supervised learning is also known as learning via examples or learning from exemplars. The following figure explains the concept.

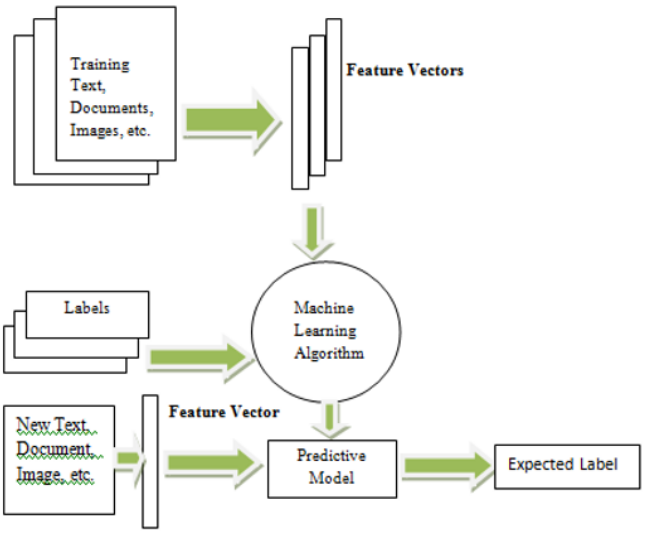

Figure: Supervised learning

Supervised learning finds applications in prediction based on historical data. For example: to predict the Iris species given a set of its flower measurements or a recognition system that determines whether an object is a galaxy, a quasar or a star given a colored image of an object through a telescope, or given an e-commerce surfing history of a person, recommendation of the products by e-commerce websites. Supervised learning tasks can be further categorized as classification tasks and regression tasks. In case of classification, the output labels are discrete whereas they are continuous in case of regression.

Unsupervised Learning

The unsupervised learning approach is all about recognizing unidentified existing patterns from the data in order to derive rules from them. This technique is appropriate in a situation when the categories of data are unknown. Here, the training data is not labeled. Unsupervised learning is regarded as a statistic based approach for learning and thus refers to the problem of finding hidden structure in unlabeled data.

Reinforcement Learning

Reinforcement learning is regarded as an intermediate type of learning as the algorithm is only provided with a response that tells whether the output is correct or not. The algorithm has to explore and rule out various possibilities to get the correct output. It is regarded as learning with a Critic as the algorithm doesn't propose any sort of suggestions or solutions to the problem.

Evolutionary Learning

It is inspired by biological organisms who adapt to their environment. The algorithm understands the behavior and adapts to the inputs and rules out unlikely solutions. It is based on the idea of fitness to propose the best solution to the problem.

Semi-Supervised Learning

These algorithms provide a technique that harnesses the power of both - supervised learning and unsupervised learning. In the previous two types output labels areeither provided for all the observations or no labels are provided. There might be situations when some observations are provided with labels but majority of observations are unlabeled due to high cost of labeling and lack of skilled human expertise. In such situations, semi-supervised algorithms are best suited for model building. Semi supervised learning can be used with problems like classification, regression and prediction.

It may further be categorized as Generative Models, Self-Training and Transductive SVM.

5.6. Ensemble Learning It is a machine learning model in which numerous learners (individual models) are trained to solve a common problem. Unlike other machine learning techniques which learn a single hypothesis from the training data, ensemble learning tries to learn by constructing a set of hypotheses from the training data and by combining them to make a prediction model [15,19] in order to decrease bias(boosting), variance (bagging), or improve predictions(stacking). Ensemble learning can be further divided into two groups:

Sequential ensemble approaches

These are the methods in which the base learners are constructed sequentially (AdaBoost). This method exploits the dependence between the base learners**.**

Parallel ensemble approaches

In these, the base learners are independent of each other, so this relationship is exploited by constructing the base learners in parallel (e.g. Random Forest)

Bagging: It stands for bootstrap aggregation. It implements homogenous learners on sample populations and takes the mean of all predictions. For example, M different trees can be trained on dissimilar subsets of data and compute the ensemble as:

$$f(x) = 1/M \sum_{m=1}^{M} f_m(x)$$

Boosting: It is an iterative technique that adjusts the observation's weight on the basis of last classification. It tries to fit a sequence of weak learner models that performs a little better than just random predicting e.g. small decision trees. AdaBoost stands for adaptive boosting and is the most widely used boosting algorithm.

Artificial Neural Network

Artificial neural networks (ANNs) are encouraged by the biological neural network. A neural network is an interconnection of neuron cells that help the electric impulses to propagate through the brain. The basic unit of learning in a neural network is a neuron, which is a nerve cell. A neuron consists of four parts, namely dendrites (receptor), soma (processor of electric signal), nucleus (core of the neuron) and axon (the transmitting end of the neuron). Analogical to a biological neural network, an ANN works on three layers: input layer, hidden layer and output layer. This type of network has weighted interconnections and learns by adjusting the weights of interconnections in order to perform parallel distributed processing. The Perceptron learning algorithm, Back-propagation algorithm, Hopfield Networks, Radial Basis Function Network (RBFN) are some popular algorithms. Based on learning behavior, ANN can be further classified as:

Supervised Neural Network

The inputs and the outputs are presented to the network as training data. The network is trained with this data by adjusting the weights to get accurate results. When it is fully trained, it is presented with unseen data to predict the output.

Unsupervised Neural Network

In unsupervised neural network, the network is not provided with any output. It tries to find some structure or correlation among the input data and group those data together in a group or class. When new data is presented to it as input, it identifies its features and classifies it in one of the groups based on similarities.

Reinforcement Neural Network

As humans interact with their environment and learn from mistakes, a reinforcement neural network also learns from its past decisions by way of penalties for wrong decisions and rewards for good decisions. The connection weights producing correct output are strengthened, while those producing incorrect responses are weakened.

Unlike other machine learning methods where clear definition of the target function are provided from the training data, this lerning method does not describe any target function in the beginning. Rather it simply stores the training instance and generalizing is postponed until a new instance is classified. Hence it is also known as lazy learner. Such methods build up a database of training instances and whenever new data is presented as input it compares that data with other instances in the database using a similarity measure to find the nearest match and make the prediction [4]. The lazy learner estimates the target function differently and locally for every new instance to be classified instead of estimating it globally for the whole instance space hence it is faster to train but, takes time in making prediction [16]. KMeans, k-medians, hierarchical clustering and expectation maximization are some popular instancebased algorithms.

Dimensionality reduction algorithms

During the past few decades, intelligent machine learning models have been adopted in numerous complex and data intensive applications like climatology, biology, astronomy, medical, economy and finance. However, existing ML based systems are not sufficiently efficient and extensible enough to deal with massive and voluminous data. High dimensionality

of data has proved to be a curse for data processing. Another challenge is sparsity of data. Global optimum is costly to find for such data. A dimensionality reduction algorithm helps in reducing the computational cost by reducing the number of dimensions of the data. It does so by reducing the redundant and irrelevant data and cleaning the data so as to improve the accuracy of results. Dimensionality reduction works in an unsupervised manner to search and exploit the implicit structure in the data [4, 5]. There are many algorithms for dimensionality reduction that can be adapted with classification and regression algorithms like Multidimensional scaling (MDS), Principal component analysis (PCA), Linear Discriminant Analysis (LDA), Principal component regression (PCR), and Linear Discriminant Analysis (LDA).

Hybrid Learning

Though ensemble learning appeared as a relief to researchers dealing with the common problems of computational complexity, over fitting and sticking to local minima in classification algorithms, researchers have found problems with ensemble learning. Complicated ensemble of multiple classifiers makes it difficult to implement and difficult to analyze the results. Instead of improving accuracy of the model, ensembles may tend to increase error at the level of individual base learner. Ensembles may result in poor accuracy as a result of selection of poor classifiers in combination. Recent approach to deal with such problems is hybridization i.e. creating ensemble of heterogeneous models. In this, more than one method is combined for example, combining clustering and decision tree or clustering and association mining etc.

Machine Learning Algorithms

In this section, we focus on some popular machine learning algorithms from the different paradigms explained in the preceding section. Although the number of algorithms falling within each paradigm are numerous and reported across pertinent literature, in this study we consider only few of these. The following table briefly explains few of these algorithms.

Supervised Learning	Decision Tree Decision	Tree is a technique for approximating discrete valued target function which represents the learnt function in the form of a decision tree

		[10]. A decision tree classifies instances by sorting them from root to some leaf nodes on the basis of feature values. Each node represents some decision (test condition) on attribute of the instance whereas every branch represents a possible value for that feature. Classification of an instance starts at the root node called the decision node. Based on the value of node, the tree traverse down along the edge which corresponds to the value of the output of feature test. This process continues in the sub-tree headed by the new node at the end of the previous edge. Finally, the leaf node signifies the classification categories or the final decision. While using a decision tree, focus is on how to decide which attribute is the best classifier at each node level. Statistical measure like information gain, Gini index, Chi-square and entropy are calculated for each node to calculate the worth of that node [10]. Several algorithms are used to implement decision trees. The most popular ones are: Classification and Regression Tree (CART), Iterative Dichotomiser 3 (ID3), Automatic Interaction Detection (CHAID), Chi-Squared C4.5 and C5.0 and M5
	Support Vector Machines	SVMs can be used for classification as well as regression problems. It is a supervised learning algorithm. It works on the concept of margin calculation. In this algorithm, each data item is plotted as a point in n-

		dimensional space (where n is the number of features we have in our dataset). The value of each feature is the value of the corresponding coordinate. It classifies the data into different classes by finding a line (hyper plane) which separates the training datasets into classes. It works by maximizing the distances between the nearest data point (in both classes) and the hyper plane that we can call as margin.
	Regression analysis	Regression analysis is a predictive modelling technique which investigates the relationship between a dependent (target) and independent variable(s) (predictor). It is an important tool for analysing and modeling of data. In this method, we try to fit the line/curve to the data points so as to minimize the differences between distances of data points from the curve or line. There are various kinds of regression analysis like linear, logistic and polynomial.
Unsupervised Learning	K-Means Clustering	K-means is a popular unsupervised machine learning algorithm for cluster analysis. Its goal is to partition 'n' observations into 'k' clusters in which each observation belongs to the cluster having the nearest mean, serving as a prototype of the cluster. The mean of the observations in a particular cluster defines the center of the cluster.
Ensemble Learning	Random Forest	It is an ensemble learning method used in classification and regression. It uses bagging approach to create a bunch of

		decision trees with random subset of data. The output of all decision trees in the random forest is combined to make the final decision trees. There are two stages in Random Forest Algorithm, one is to create random forest, and the other is to make a prediction from the random forest classifier created in the first stage.
Dimensionality Reduction	Principal Component Algorithm	It is primarily used for reducing dimensionality of data set. It helps in reducing the number of features of the data set or the number of independent variables in the data set. It uses orthogonal transformation to convert correlated variables into a set of linearly uncorrelated variables called principal components.

Literature Review:

Yannakoudakis et al. (2011) developed corpora that contain 1244 essays and ten prompts. This corpus evaluates whether a student can write the relevant English sentences without any grammatical and spelling mistakes. This type of corpus helps to test the models built for GRE and TOFEL type of exams. It gives scores between 1 and 40. Bailey and Meurers (2008), Created a dataset (CREE reading comprehension) for language learners and automated short answer scoring systems. The corpus consists of 566 responses from intermediate students. Mohler and Mihalcea (2009). Created a dataset for the computer science domain consists of 630 responses for data structure assignment questions. The scores are range from 0 to 5 given by two human raters. Dzikovska et al. (2012) created a Student Response Analysis (SRA) corpus. It consists of two sub-groups: the BEETLE corpus consists of 56 questions and approximately 3000 responses from students in the electrical and electronics domain. The second one is the SCIENTSBANK (SemEval-2013) (Dzikovska et al. 2013a; b) corpus consists of 10,000

responses on 197 prompts on various science domains. The student response ladled with "correct, partially correct incomplete, Contradictory, Irrelevant, Non-domain.

In the Kaggle (2012) competition, released total 3 types of corpuses on an Automated Student Assessment Prize (ASAP1) (“https:// www. kaggle. com/c/ asap- sas/”) essays and short answers. It has nearly 17,450 essays, out of which it provides up to 3000 essays for each prompt. It has eight prompts that test 7th to 10th grade US students. It gives scores between the [0–3] and [0–60] range. The limitations of these corpora are: (1) it has a different score range for other prompts. (2) It uses statistical features such as named entities extraction and lexical features of words to evaluate essays. ASAP ++ is one more dataset from Kaggle. It is with six prompts, and each prompt has more than 1000 responses total of 10,696 from 8th-grade students. Another corpus contains ten prompts from science, English domains and a total of 17,207 responses. Two human graders evaluated all these responses. Correnti et al. (2013) created a Response-to-Text Assessment (RTA) dataset used to check student writing skills in all directions like style, mechanism, and organization. 4–8 grade students give the responses to RTA. Basu et al. (2013) created a power grading dataset with 700 responses for ten different prompts from US immigration exams. It contains all short answers for assessment. The TOEFL11 corpus Blanchard et al. (2013) contains 1100 essays evenly distributed over eight prompts. It is used to test the English language skills of a candidate attending the TOFEL exam. It scores the language proficiency of a candidate as low, medium, and high. International Corpus of Learner English (ICLE) Granger et al. (2009) built a corpus of 3663 essays covering different dimensions. It has 12 prompts with 1003 essays that test the organizational skill of essay writing, and13 prompts, each with 830 essays that examine the thesis clarity and prompt adherence. Argument Annotated Essays (AAE) Stab and Gurevych (2014) developed a corpus that contains 102 essays with 101 prompts taken from the essayforum2 site. It tests the persuasive nature of the student essay. The SCIENTSBANK corpus used by Sakaguchi et al. (2015) available in git-hub, containing 9804 answers to 197 questions in 15 science domains. Table 3 illustrates all datasets related to AES systems.

Regression based methods

The goal of the regression task is to predict the score of an essay. The classification task is to classify the essays belonging to (low, medium, or highly) relevant to the question's topic. Since the last three years, most AES systems developed made use of the concept of the neural network.

Mohler and Mihalcea (2009). proposed text-to-text semantic similarity to assign a score to the student essays. There are two text similarity measures like Knowledge-based measures, corpus-based measures. There eight knowledge-based tests with all eight models. They found the similarity. The shortest path similarity determines based on the length, which shortest path between two contexts. Leacock & Chodorow find the similarity based on the shortest path's length between two concepts using node-counting. The Lesk similarity finds the overlap between the corresponding definitions, and Wu & Palmer algorithm finds similarities based on the depth of two given concepts in the wordnet taxonomy. Resnik, Lin, Jiang&Conrath, Hirst& St-Onge find the similarity based on different parameters like the concept, probability, normalization factor, lexical chains. In corpus-based likeness, there LSA BNC, LSA Wikipedia, and ESA Wikipedia, latent semantic analysis is trained on Wikipedia and has excellent domain knowledge. Among all similarity scores, correlation scores LSA Wikipedia scoring accuracy is more. But these similarity measure algorithms are not using NLP concepts. These models are before 2010 and basic concept models to continue the research automated essay grading with updated algorithms on neural networks with content-based features. Adamson et al. (2014) proposed an automatic essay grading system which is a statisticalbased approach in this they retrieved features like POS, Character count, Word count, Sentence count, Miss spelled words, n-gram representation of words to prepare essay vector. They formed a matrix with these all vectors in that they applied LSA to give a score to each essay. It is a statistical approach that doesn't consider the semantics of the essay. The accuracy they got when compared to the human rater score with the system is 0.532. Cummins et al. (2016). Proposed Timed Aggregate Perceptron vector model to give ranking to all the essays, and later they converted the rank algorithm to predict the score of the essay. The model trained with features like Word unigrams, bigrams, POS, Essay length, grammatical relation, Max word length, sentence length. It is multi-task learning, gives ranking to the essays, and predicts the score for the essay. The performance evaluated through QWK is 0.69, a substantial agreement between the human rater and the system. Sultan et al. (2016). Proposed a Ridge regression model to find short answer scoring with Question Demoting. Question Demoting is the new concept included in the essay's final

assessment to eliminate duplicate words from the essay. The extracted features are Text Similarity, which is the similarity between the student response and reference answer. Question Demoting is the number of repeats in a student response. With inverse document frequency, they assigned term weight. The sentence length Ratio is the number of words in the student response, is another feature. With these features, the Ridge regression model was used, and the accuracy they got 0.887. Contreras et al. (2018). Proposed Ontology based on text mining in this model has given a score for essays in phases. In phase-I, they generated ontologies with ontoGen and SVM to find the concept and similarity in the essay. In phase II from ontologies, they retrieved features like essay length, word counts, correctness, vocabulary, and types of word used, domain information. After retrieving statistical data, they used a linear regression model to find the score of the essay. The accuracy score is the average of 0.5. Darwish and Mohamed (2020) proposed the fusion of fuzzy Ontology with LSA. They retrieve two types of features, like syntax features and semantic features. In syntax features, they found Lexical Analysis with tokens, and they construct a parse tree. If the parse tree is broken, the essay is inconsistent—a separate grade assigned to the essay concerning syntax features. The semantic features are like similarity analysis, Spatial Data Analysis. Similarity analysis is to find duplicate sentences—Spatial Data Analysis for finding Euclid distance between the center and part. Later they combine syntax features and morphological features score for the final score. The accuracy they achieved with the multiple linear regression model is 0.77, mostly on statistical features. Süzen Neslihan et al. (2020) proposed a text mining approach for short answer grading. First, their comparing model answers with student response by calculating the distance between two sentences. By comparing the model answer with student response, they find the essay's completeness and provide feedback. In this approach, model vocabulary plays a vital role in grading, and with this model vocabulary, the grade will be assigned to the student's response and provides feedback. The correlation between the student answer to model answer is 0.81.

Classification based Models

Persing and Ng (2013) used a support vector machine to score the essay. The features extracted are OS, N-gram, and semantic text to train the model and identified the keywords from the essay to give the final score. Sakaguchi et al. (2015) proposed two methods: response-based and reference-based. In response-based scoring, the extracted features are response length, n-gram

model, and syntactic elements to train the support vector regression model. In reference-based scoring, features such as sentence similarity using word2vec is used to find the cosine similarity of the sentences that is the final score of the response. First, the scores were discovered individually and later combined two features to find a final score. This system gave a remarkable increase in performance by combining the scores. Mathias and Bhattacharyya (2018a; b) Proposed Automated Essay Grading Dataset with Essay Attribute Scores. The first concept features selection depends on the essay type. So the common attributes are Content, Organization, Word Choice, Sentence Fluency, Conventions. In this system, each attribute is scored individually, with the strength of each attribute identified. The model they used is a random forest classifier to assign scores to individual attributes. The accuracy they got with QWK is 0.74 for prompt 1 of the ASAS dataset (https:// www. kaggle. com/c/ asap- sas/). Ke et al. (2019) used a support vector machine to find the response score. In this method, features like Agreeability, Specificity, Clarity, Relevance to prompt, Conciseness, Eloquence, Confidence, Direction of development, Justification of opinion, and Justification of importance. First, the individual parameter score obtained was later combined with all scores to give a final response score. The features are used in the neural network to find whether the sentence is relevant to the topic or not. Salim et al. (2019) proposed an XGBoost Machine Learning classifier to assess the essays. The algorithm trained on features like word count, POS, parse tree depth, and coherence in the articles with sentence similarity percentage; cohesion and coherence are considered for training. And they implemented K-fold cross-validation for a result the average accuracy after specific validations is 68.12.

Neural network models

Shehab et al. (2016) proposed a neural network method that used learning vector quantization to train human scored essays. After training, the network can provide a score to the ungraded essays. First, we should process the essay to remove Spell checking and then perform preprocessing steps like Document Tokenization, stop word removal, Stemming, and submit it to the neural network. Finally, the model will provide feedback on the essay, whether it is relevant to the topic. And the correlation coefficient between human rater and system score is 0.7665. Kopparapu and De (2016) proposed the Automatic Ranking of Essays using Structural and Semantic Features. This approach constructed a super essay with all the responses. Next, ranking

for a student essay is done based on the super-essay. The structural and semantic features derived helps to obtain the scores. In a paragraph, 15 Structural features like an average number of sentences, the average length of sentences, and the count of words, nouns, verbs, adjectives, etc., are used to obtain a syntactic score. A similarity score is used as semantic features to calculate the overall score. Dong and Zhang (2016) proposed a hierarchical CNN model. The model builds two layers with word embedding to represents the words as the first layer. The second layer is a word convolution layer with max-pooling to find word vectors. The next layer is a sentence-level convolution layer with max-pooling to find the sentence's content and synonyms. A fully connected dense layer produces an output score for an essay. The accuracy with the hierarchical CNN model resulted in an average QWK of 0.754. Taghipour and Ng (2016) proposed a first neural approach for essay scoring build in which convolution and recurrent neural network concepts help in scoring an essay. The network uses a lookup table with the one-hot representation of the word vector of an essay. The final efficiency of the network model with LSTM resulted in an average QWK of 0.708. Dong et al. (2017). Proposed an Attention-based scoring system with CNN + LSTM to score an essay. For CNN, the input parameters were character embedding and word embedding, and it has attention pooling layers and used NLTK to obtain word and character embedding. The output gives a sentence vector, which provides sentence weight. After CNN, it will have an LSTM layer with an attention pooling layer, and this final layer results in the final score of the responses. The average QWK score is 0.764. Riordan et al. (2017) proposed a neural network with CNN and LSTM layers. Word embedding, given as input to a neural network. An LSTM network layer will retrieve the window features and delivers them to the aggregation layer. The aggregation layer is a superficial layer that takes a correct window of words and gives successive layers to predict the answer's sore. The accuracy of the neural network resulted in a QWK of 0.90. Zhao et al. (2017) proposed a new concept called Memory-Augmented Neural network with four layers, input representation layer, memory addressing layer, memory reading layer, and output layer. An input layer represents all essays in a vector form based on essay length. After converting the word vector, the memory addressing layer takes a sample of the essay and weighs all the terms. The memory reading layer takes the input from memory addressing segment and finds the content to finalize the score. Finally, the output layer will provide the final score of the essay. The accuracy of essay scores is 0.78, which is far better than the LSTM neural network. Mathias and Bhattacharyya (2018a; b) proposed

deep learning networks using LSTM with the CNN layer and GloVe pre-trained word embeddings. For this, they retrieved features like Sentence count essays, word count per sentence, Number of OOVs in the sentence, Language model score, and the text's perplexity. The network predicted the goodness scores of each essay. The higher the goodness scores, means higher the rank and vice versa. Nguyen and Dery (2016). Proposed Neural Networks for Automated Essay Grading. In this method, a single layer bi-directional LSTM accepting word vector as input. Glove vectors used in this method resulted in an accuracy of 90%. Ruseti et al. (2018) proposed a recurrent neural network that is capable of memorizing the text and generate a summary of an essay. The Bi-GRU network with the maxpooling layer molded on the word embedding of each document. It will provide scoring to the essay by comparing it with a summary of the essay from another Bi-GRU network. The result obtained an accuracy of 0.55. Wang et al. (2018a; b) proposed an automatic scoring system with the bi-LSTM recurrent neural network model and retrieved the features using the word2vec technique. This method generated word embeddings from the essay words using the skip-gram model. And later, word embedding is used to train the neural network to find the final score. The softmax layer in LSTM obtains the importance of each word. This method used a QWK score of 0.83%. Dasgupta et al. (2018) proposed a technique for essay scoring with augmenting textual qualitative Features. It extracted three types of linguistic, cognitive, and psychological features associated with a text document. The linguistic features are Part of Speech (POS), Universal Dependency relations, Structural Well-formedness, Lexical Diversity, Sentence Cohesion, Causality, and Informativeness of the text. The psychological features derived from the Linguistic Information and Word Count (LIWC) tool. They implemented a convolution recurrent neural network that takes input as word embedding and sentence vector, retrieved from the GloVe word vector. And the second layer is the Convolution Layer to find local features. The next layer is the recurrent neural network (LSTM) to find corresponding of the text. The accuracy of this method resulted in an average QWK of 0.764. Liang et al. (2018) proposed a symmetrical neural network AES model with Bi-LSTM. They are extracting features from sample essays and student essays and preparing an embedding layer as input. The embedding layer output is transfer to the convolution layer from that LSTM will be trained. Hear the LSRM model has self-features extraction layer, which will find the essay's coherence. The average QWK score of SBLSTMA is 0.801. Liu et al. (2019) proposed two-stage learning. In the first stage, they are assigning a score based on semantic data

from the essay. The second stage scoring is based on some handcrafted features like grammar correction, essay length, number of sentences, etc. The average score of the two stages is 0.709. Pedro Uria Rodriguez et al. (2019) proposed a sequence-to-sequence learning model for automatic essay scoring. They used BERT (Bidirectional Encoder Representations from Transformers), which extracts the semantics from a sentence from both directions. And XLnet sequence to sequence learning model to extract features like the next sentence in an essay. With this pre-trained model, they attained coherence from the essay to give the final score. The average QWK score of the model is 75.5. Xia et al. (2019) proposed a two-layer Bi-directional LSTM neural network for the scoring of essays. The features extracted with word2vec to train the LSTM and accuracy of the model in an average of QWK is 0.870. Kumar et al. (2019) Proposed an AutoSAS for short answer scoring. It used pre-trained Word2Vec and Doc2Vec models trained on Google News corpus and Wikipedia dump, respectively, to retrieve the features. First, they tagged every word POS and they found weighted words from the response. It also found prompt overlap to observe how the answer is relevant to the topic, and they defined lexical overlaps like noun overlap, argument overlap, and content overlap. This method used some statistical features like word frequency, difficulty, diversity, number of unique words in each response, type-token ratio, statistics of the sentence, word length, and logical operator-based features. This method uses a random forest model to train the dataset. The data set has sample responses with their associated score. The model will retrieve the features from both responses like graded and ungraded short answers with questions. The accuracy of AutoSAS with QWK is 0.78. It will work on any topics like Science, Arts, Biology, and English. Jiaqi Lun et al. (2020) proposed an automatic short answer scoring with BERT. In this with a reference answer comparing student responses and assigning scores. The data augmentation is done with a neural network and with one correct answer from the dataset classifying reaming responses as correct or incorrect. Zhu and Sun (2020) proposed a multimodal Machine Learning approach for automated essay scoring. First, they count the grammar score with the spaCy library and numerical count as the number of words and sentences with the same library. With this input, they trained a single and Bi LSTM neural network for finding the final score. For the LSTM model, they prepared sentence vectors with GloVe and word embedding with NLTK. BiLSTM will check each sentence in both directions to find semantic from the essay. The average QWK score with multiple models is 0.70.

Ontology based approach

Mohler et al. (2011) proposed a graph-based method to find semantic similarity in short answer scoring. For the ranking of answers, they used the support vector regression model. The bag of words is the main feature extracted in the system. Ramachandran et al. (2015) also proposed a graph-based approach to find lexical based semantics. Identified phrase patterns and text patterns are the features to train a random forest regression model to score the essays. The accuracy of the model in a QWK is 0.78. Zupanc et al. (2017) proposed sentence similarity networks to find the essay's score. Ajetunmobi and Daramola (2017) recommended an ontology-based information extraction approach and domain-based ontology to find the score.

Speech response scoring

Automatic scoring is in two ways one is text-based scoring, other is speech-based scoring. This paper discussed text-based scoring and its challenges, and now we cover speech scoring and common points between text and speech-based scoring. Evanini and Wang (2013), Worked on speech scoring of non-native school students, extracted features with speech ratter, and trained a linear regression model, concluding that accuracy varies based on voice pitching. Loukina et al. (2015) worked on feature selection from speech data and trained. SVM. Malinin et al. (2016) used neural network models to train the data. Loukina et al. (2017). Proposed speech and text-based automatic scoring. Extracted text-based features, speech-based features and trained a deep neural network for speech-based scoring. They extracted 33 types of features based on acoustic signals. Malinin et al. (2017). Wu Xixin et al. (2020) Worked on deep neural networks for spoken language assessment. Incorporated different types of models and tested them. Ramanarayanan et al. (2017) worked on feature extraction methods and extracted punctuation, fluency, and stress and trained different Machine Learning models for scoring. Knill et al. (2018). Worked on Automatic speech recognizer and its errors how its impacts the speech assessment.

AES approaches into three categories. Regression models, classification models, and neural network models. The regression models failed to find cohesion and coherence from the essay because it trained on BoW(Bag of Words) features. In processing data from input to output, the regression models are less complicated than neural networks. There are unable to find many

intricate patterns from the essay and unable to find sentence connectivity. If we train the model with BoW features in the neural network approach, the model never considers the essay's coherence and coherence. First, to train a Machine Learning algorithm with essays, all the essays are converted to vector form. We can form a vector with BoW and Word2vec, TF-IDF. The BoW and Word2vec vector representation of essays represented in Table 6. The vector representation of BoW with TF-IDF is not incorporating the essays semantic, and it's just statistical learning from a given vector. Word2vec vector comprises semantic of essay in a unidirectional way. In BoW, the vector contains the frequency of word occurrences in the essay. The vector represents 1 and more based on the happenings of words in the essay and 0 for not present. So, in BoW, the vector does not maintain the relationship with adjacent words; it's just for single words. In word2vec, the vector represents the relationship between words with other words and sentences prompt in multiple dimensional ways. But word2vec prepares vectors in a unidirectional way, not in a bidirectional way; word2vec fails to find semantic vectors when a word has two meanings, and the meaning depends on adjacent words. Table 7 represents a comparison of Machine Learning models and features extracting methods. In AES, cohesion and coherence will check the content of the essay concerning the essay prompt these can be extracted from essay in the vector from. Two more parameters are there to access an essay is completeness and feedback. Completeness will check whether student's response is sufficient or not though the student wrote correctly.

Students' feedback is an effective tool that provides valuable insights concerning various educational entities including teachers, courses, institutions, etc. and teaching aspects related to these entities. The identification of these aspects as expressed in the textual comments of students is of great importance as it aids decision makers to take the right action to specifically improve them. In this context, we examined and classified the reviewed papers based on the aspects that concerned students and that the authors aimed to investigate. In particular, we found three categories and their related teaching aspects which were objects of investigation in these papers: the first category comprised studies dealing with the comments of students concerning various aspects of the teacher entity, including the teacher's knowledge, pedagogy, behavior, etc; the second category contained papers concerning various aspects of the three different entities, such as courses, teachers, and institutions. Course-related aspects included dimensions such as course content, course structure, assessment, etc., whereas aspects associated to the institution

entity were tuition fees, the campus, student life, etc.; the third category included papers dealing with capturing the opinions and attitudes of students toward institution entities. Python-based NLP and machine learning packages, libraries, and tools (colored in blue) are among the most popular solutions due to the open-source nature of the Python programming language. Specifically, the NLTK (Natural Language Toolkit) package is the dominant solution, and it was used in 12 different articles for pre-processing tasks including tokenizing, part-of-speech, normalization, the cleaning of text, etc. Java-based NLP and machine learning packages, frameworks, libraries, and tools constitute the second group of solutions used for sentiment analysis. Rapidminer is the most common Java-based framework and was used in three articles. The third group is composed of NLP and machine learning solutions based on the R programming language. Only three studies used solutions in this group to conduct the sentiment analysis task.

Extracting and Generating Text with Part-of-Speech Tags

Almost every NLP application needs to extract specific information from a text and generate new text that is relevant to a particular situation. For example, a chatbot must be able carry on a conversation with a user, which means it must be able to identify specific parts of a user's text and then generate its own appropriate response. Let's look at how to do all of those using linguistic features.

Part-of-speech tags can help you retrieve specific kinds of information from a text, and they can also help you generate entirely new sentences based on a submitted one.

Numeric, Symbolic, and Punctuation Tags In addition to part-of-speech tags for nouns, verbs, and other words in a sentence, spaCy has tags for symbols, numbers, and punctuation marks. Let's look at these by processing the following sentence:

Elon Musk earned $200 Billion.

To begin, let's extract the coarse-grained part-of-speech features from the tokens in the sentence to see how spaCy distinguishes between different part-of-speech categories. We can do this with the following script:

```
>>> import spacy
>>> nlp = spacy.load('en')
>>> doc = nlp(u"The firm earned $1.5 million in 2017.")
>>> for token in doc:
...   print(token.text, ❶token.pos_, ❷spacy.explain(token.pos_))
```

We create a Doc object for the submitted sentence and then output the coarse-grained part-of-speech tags ❶. We also use the spacy.explain() function, which returns a description for a given linguistic feature ❷.

```
The    DET   determiner
firm   NOUN  noun
earned VERB  verb
$      SYM   symbol
1.5    NUM   numeral
million NUM  numeral
in     ADP   adposition
2017   NUM   numeral
.      PUNCT punctuation
```

Now, for the sake of comparison, let's output both coarsegrained and fine-grained part-of-speech tags for this sample sentence along with a description column for the fine-grained tags:

```
>>> for token in doc:
...   print(token.text, token.pos_, token.tag_, spacy.explain(token.tag_))
```

The output should look as follows:

```
The    DET   DT  determiner
firm   NOUN  NN  noun, singular
earned VERB  VBD verb, past tense
$      SYM   $   symbol, currency
1.5    NUM   CD  cardinal number
million NUM  CD  cardinal number
in     ADP   IN  conjunction, subordinating or preposition
2017   NUM   CD  cardinal number
.      PUNCT .   punctuation mark, sentence closer
```

The second and third columns contain the coarse-grained and fine-grained part-of-speech tags, respectively. The fourth column gives descriptions of the fine-grained tags provided in the third column. The fine-grained tagging divides each category into subcategories. For example, the coarse-grained category SYM (symbols) has three fine-grained subcategories. These are $ for currency symbols, # for the number sign, and SYM for all the other symbols, such as +, −, ×, ÷, =. This sub-dividing can be useful when you need to distinguish between different types of symbols. For example, you might be processing articles about math and want your script to recognize symbols commonly found in math formulas. Or you might be writing a script that needs to recognize currency symbols in financial reports.

Turning Statements into Questions

Suppose your NLP application must be able to generate a question from a submitted statement. For example, one way chatbots maintain conversations with the user is by asking the user a confirmatory question. When a user says, “I am sure,” the chatbot might ask something like, “Are you really sure?” To do this, the chatbot must be able to generate a relevant question. Let’s say the user’s submitted sentence is this:

I can promise it is worth your time.

This sentence contains several verbs and pronouns, each with different morphologies. To see this more clearly, let’s look at the part-of-speech tags spaCy assigned to the tokens in this sentence:

```
>>> doc = nlp(u"I can promise it is worth your time.")
>>> for token in doc:
...   print(token.text, token.pos_, token.tag_)
...
```

We print the tokens, their coarse-grained part-of-speech tags, and their fine-grained part-of-speech-tags, producing the following output:

```
I      PRON PRP
can    VERB MD
promise VERB VB
it     PRON PRP
is     VERB VBZ
worth  ADJ  JJ
your   ADJ  PRP$
time   NOUN NN
.      PUNCT .
```

From the fine-grained part-of-speech tags, you can distinguish between the morphological categories of the verbs and pronouns present in the sentence. For example, the finegrained part-of-speech tag PRP marks personal pronouns and PRP$ marks possessive pronouns, allowing you to distinguish between these two types of pronouns programmatically. We'll need this information when working on this example. A confirmatory question to the sentence discussed here might be as follows (another statement would require another confirmatory question, of course):

Can you really promise it is worth my time?

From a human perspective, forming this question from the statement looks pretty straightforward: you change the order of some words, alter the pronouns accordingly, and add the adverbial modifier "really" to the main verb (the one that comes right after the subject). But how can you accomplish all these operations programmatically? Let's look at some part-of-speech tags. In the sample sentence, the verbs involved in forming the question are "can" and "promise". The fine-grained part-of-speech tags mark the first one, "can", as a modal auxiliary verb and the second one as a verb in the base form. Notice that in the preceding confirmatory

question, the modal auxiliary verb has switched places with the personal pronoun, a process called inversion. We'll have to implement this in the script. When it comes to the pronouns, the chatbot should follow a pattern common to regular conversations. Table 4-1 summarizes the use of pronouns in such an application.

The following steps outline what we need to do to generate a question from the original statement:

1. Change the order of words in the original sentence from "subject + modal auxiliary verb + infinitive verb" to "modal auxiliary verb + subject + infinitive verb."
2. Replace the personal pronoun "I" (the sentence's subject) with "you."
3. Replace the possessive pronoun "your" with "my."
4. Place the adverbial modifier "really" before the verb "promise" to emphasize the latter.
5. Replace the punctuation mark "." with "?" at the end of the sentence.

The following script implements these steps:

```
import spacy
nlp = spacy.load('en')
doc = nlp(u"I can promise it is worth your time.")
sent = ''
for i,token in enumerate(doc):
❶ if token.tag_ == 'PRP' and doc[i+1].tag_ == 'MD' and doc[i+2].tag_ == 'VB':
  ❷ sent = doc[i+1].text.capitalize() + ' ' + doc[i].text
    sent = sent + ' ' + ❸doc[i+2:].text
  ❹ break

#By now, you should have: 'Can I promise it is worth your time.'
 #Retokenization
❼ doc=nlp(sent)
 for i,token in enumerate(doc):
❻ if token.tag_ == 'PRP' and token.text == 'I':
    sent = doc[:i].text + ' you ' + doc[i+1:].text
    break
#By now, you should have: 'Can you promise it is worth your time.'
doc=nlp(sent)
for i,token in enumerate(doc):
❼ if token.tag_ == 'PRP$' and token.text == 'your':
    sent = doc[:i].text + ' my ' + doc[i+1:].text
    break
#By now, you should have: 'Can you promise it is worth my time.'
doc=nlp(sent)
for i,token in enumerate(doc):
 if token.tag_ == 'VB':
  ❽ sent = doc[:i].text + ' really ' + doc[i:].text
   break
#By now, you should have: 'Can you really promise it is worth my time.'
doc=nlp(sent)
❾ sent = doc[:len(doc)-1].text + '?'
#Finally, you should have: 'Can you really promise it is worth my time?'
```

We perform the first four steps in separate for loops. First, we iterate over the tokens in the sentence and change the order of the subject and verb to make the sentence a question. In this example, we're looking for the modal auxiliary verb (tagged MD) that follows a personal pronoun and is followed by an infinitive verb ❶. Once we find this sequence of words, we move the modal auxiliary verb immediately before the personal pronoun, placing it at the beginning of the sentence ❷. To compose a new sentence, we use a technique known in Python as slicing that allows us to extract a subsequence from a sequence object, such as a string or a list, by specifying the start and end indices. In this case, we can apply slicing to a Doc object to extract a given subsequence of tokens from it. For example, slice doc[2:] will contain the doc's tokens starting from the token at index 2 through the end of the doc, which in this case, is "promise it is

worth your time.” ❸. Once we move the modal verb to a new position, we exit the for loop ❹. You might wonder why we don’t just use the personal pronoun and auxiliary modal verb’s indices to perform inversion. Because we know the personal pronoun is at index 0 and the

modal verb is at index 1, why do we have to use a loop that iterates over the entire set of tokens to find the modal verb’s position? Won’t the verb always follow the subject and so be the second word in the sentence? The fact is that a sentence doesn’t always start with the subject. For example, what if the sentence were “Sure enough, I can promise it is worth your time.”? In that case, the script would know to omit the first two words and start processing with the subject. As a result of the inversion, we get the new sentence as a string. To make this sentence available for further processing, we need to obtain a Doc object for it ❺. Next, we create a new for loop that will replace the personal pronoun “I” with the personal pronoun “you.” To do this, we search for personal pronouns (tagged PRP). If the personal pronoun is “I,” we replace it with “you” ❻. Then we quit the for loop. We repeat this process to replace the possessive pronoun “your” with “my” by searching for the PRP$ tag ❼. Then, in a new for loop, we find a verb in the infinitive form and insert the adverbial modifier “really” before it ❽. Finally, we replace the sentence’s period with a question mark. This is the only step where we don’t need to use a loop. The reason is that in all possible sentences, the period and the question mark go at the end of a sentence, so we can reliably find them using their indices with len(doc)-1

Can you really promise it is worth my time?

This script is a good start, but it won’t work with every submitted statement. For example, the statement might contain a personal pronoun other than “I,” but our script doesn’t explicitly check for that. Also, some sentences don’t contain auxiliary verbs, like the sentence “I love eating ice cream.” In those cases, we’d have to use the word “do” to form the question instead of a word like “can” or “should,” like this: “Do you really love eating ice cream?” But if the sentence contains the verb,

“to be,” as in the sentence “I am sleepy,” we’d have to move that verb to the front, like this: “Are you sleepy?” A real implementation of this chatbot would have to be able to choose the appropriate option for a submitted sentence. You’ll see a “do” example in “Deciding.

Understanding word vectors

Word vectors are the series of real numbers that represent the meanings of natural language words. When building statistical models, we map words to vectors of real numbers that reflect the words' semantic similarity. You can imagine a word vector space as a cloud in which the vectors of words with similar meanings are located nearby. For instance, he vector representing the word "potato" should be closer to the vector of the word "carrot" than to that of the word "crying." To generate these vectors, we must be able to encode the meaning of these words. There are a few approaches to encoding meaning, which we'll outline in this section.

	Country	Capital	Greek	Italian
Italy	1	0	0	1
Rome	0	1	0	1
Greece	1	0	1	0
Athens	0	1	1	0

We've distributed the meaning of each word between its coordinates in a four-dimensional space, representing the categories "Country," "Capital," "Greek," and "Italian." In this simplified example, a coordinate value can be either 1 or 0, indicating whether or not a corresponding word belongs to the category. Once you have a vector space in which vectors of numbers capture the meaning of corresponding words, you can use vector arithmetic on this vector space to gain insight into a word's meaning. To find out which country Athens is the capital of, you could use the following equation, where each token stands for its corresponding vector and X is an unknown vector:

Italy - Rome = X – Athens

This equation expresses an analogy in which X represents the word vector that has the same relationship to Athens as Italy has to Rome. To solve for X, we can rewrite the equation like this:

X = Italy - Rome + Athens

We first subtract the vector Rome from the vector Italy by subtracting the corresponding vector elements. Then we add the sum of the resulting vector and the vector Athens.

Using Dimensions to Represent Meaning

Although the vector space we just created had only four categories, a real-world vector space might require tens of thousands. A vector space of this size would be impractical for most applications, because it would require a huge wordembedding matrix. For example, if you had 10,000 categories and 1,000,000 entities to encode, you'd need a 10,000 × 1,000,000 embedding matrix, making operations on it too timeconsuming. The obvious approach to reducing the size of the embedding matrix is to reduce the number of categories in the vector space. Instead of using coordinates to represent all categories, a real-world implementation of a word vector space uses the distance between vectors to quantify and categorize semantic similarities. The individual dimensions typically don't have inherent meanings. Instead, they represent locations in the vector space, and the distance between vectors indicates the similarity of the corresponding words' meanings. The following is a fragment of the 300-dimensional word vector space extracted from the fastText, a word vector library, which you can download at https://fasttext.cc/docs/en/englishvectors.html:

```
compete   -0.0535 -0.0207 0.0574 0.0562 ... -0.0389 -0.0389
equations -0.0337 0.2013 -0.1587 0.1499 ...  0.1504 0.1151
Upper     -0.1132 -0.0927 0.1991 -0.0302 ... -0.1209 0.2132
mentor     0.0397 0.1639 0.1005 -0.1420 ... -0.2076 -0.0238
reviewer  -0.0424 -0.0304 -0.0031 0.0874 ... 0.1403 -0.0258
```

Each line contains a word represented as a vector of real numbers in multidimensional space. Graphically, we can represent a 300-dimensional vector space like this one with either a 2D or 3D projection. To prepare such a projection, we can use first two or three principal coordinates of a vector, respectively. Figure 5-1 shows vectors from a 300-dimensional vector space in a 2D projection.

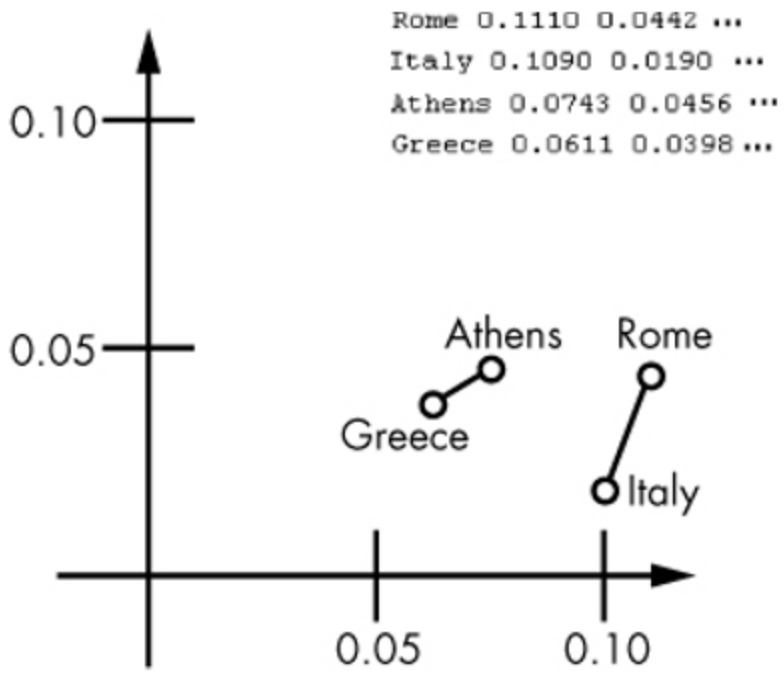

Figure : A fragment of a 2D projection of a multidimensional vector space

One interesting detail you might notice here is that the lines connecting Greece with Athens and Italy with Rome, respectively, are almost parallel. Their lengths also look comparable. In practice, this means that if you have three out of the above four vectors, you can calculate an approximate location of the missing one, since you know where to shift the vector and how far. The vectors in the diagram illustrate a country-capital relation, but they could easily have another type of relation, such as male-female, verb tense, and so on.

Using Semantic Similarity for Categorization Tasks

Determining two objects' syntactic similarity can help you sort texts into categories or pick out only the relevant texts. For example, suppose you're sorting through user comments posted to a website to find all the comments related to the word "fruits." Let's say you have the following utterances to evaluate:

I want to buy this beautiful book at the end of the week.
Sales of citrus have increased over the last year.\
How much do you know about this type of tree?

You can easily recognize that only the second sentence is directly related to fruits because it contains the word "citrus." But to pick out this sentence programmatically, you'll have to compare the word vector for the word "fruits" with word vectors in the sample sentences. Let's start with the simplest but least successful way of doing this task: comparing "fruits" to each of the sentences. As stated earlier, spaCy determines the similarity of two container objects by comparing their corresponding word vectors. To compare a single token with an entire sentence, spaCy averages the sentence's word vectors to generate an entirely new vector. The following script compares each of the preceding sentence samples with the word "fruits":

```
import spacy
nlp = spacy.load('en')
❶ token = nlp(u'fruits')[0]
❷ doc = nlp(u'I want to buy this beautiful book at the end of the week. Sales of
citrus have increased over the last year. How much do you know about this type
of tree?')
❸ for sent in doc.sents:
    print(sent.text)
❹ print('similarity to', token.text, 'is', token.similarity(sent),'\n')
```

We first create a Token object for the word "fruits" ❶. Then we apply the pipeline to the sentences we're categorizing, creating a single Doc object to hold all of them ❷. We shred the doc into sentences ❸, and then print each of the sentences and their semantic similarity to the token "fruits," which we acquire using the token object's similarity method ❹.

The output should look something like this (although the actual figures will depend on the model you use):

```
I want to buy this beautiful book at the end of the week.
similarity to fruits is 0.06307832979619851
Sales of citrus have increased over the last year.
similarity to fruits is 0.2712141843864381
How much do you know about this type of tree?
similarity to fruits is 0.24646341651210604
```

The degree of similarity between the word "fruits" and the first sentence is very small, indicating that the sentence has nothing to do with fruits. The second sentence—the one that includes the word "citrus"—is the most closely related to "fruits," meaning the script correctly identified the

relevant sentence. But notice that the script also identified the third sentence as being somehow related to fruits, probably because it includes the word “tree,” and fruits grow on trees. It would be naive to think that the similarity measuring algorithm “knows” that orange and citrus are fruits. All it knows is that these words (“orange” and “citrus”) often share the same context with word “fruit” and therefore they’ve been put close to it in the vector space. But the word “tree” can also often be found in the same context as the word “fruit.” For example, the phrase “fruit tree” is not uncommon. For that reason the level of similarity calculated between “fruits” (or “fruit” as its lemma) and “tree” is close to the result we got for “citrus” and “fruits.” There’s another problem with this approach to categorizing texts. In practice, of course, you might sometimes have to deal with texts that are much larger than the sample texts used in this section. If the text you’re averaging is very large, the most important words might have little to no effect on the syntactic similarity value.