Genetic Programming for Sentiment Analysis

Nathan Kaffes

***Abstract*—Sentiment Analysis is a complicated problem relating to the representation of text as data and the overall garnering of quantifiable information from the sentiment embedded within this text. This project aims to develop a Genetic Programming with Symbolic Regression model for the use of application to the problem of Sentiment Analysis, which is a Natural Language Processing (NLP) problem that requires high levels of preprocessing and data transmutation. Sentiment Analysis is an important task for the use of opinion farming and response evaluation. Solving this issue with a Genetic Program could help to further the area of expertise surrounding NLP problems and optimise the efficiency of solving said problems. The model created is a Genetic Programming model with Symbolic Regression techniques implemented for fitness evaluation. Multiple datasets are processed and used for testing. The validity and efficiency of the model is compared against existing models currently circulating and the results of such experiments define the success of this project. The key result from this analysis and project will be the sufficiency for the purpose that a Genetic Programming model brings to the NLP space of Sentiment Analysis models, and the interpretability of the model comparatively to other existing examples.**

I. Introduction

Sentiment Analysis is the task of taking text and transmuting it in a way that a model can detect the sentiment within it. This task requires large amounts of preprocessing and feature engineering to fully represent the trends within language. Sentiment Analysis has existed within the public sphere for decades. Businesses rely on the sentiment of their customers and without proper, trustworthy responses, many businesses fumble around in the dark attempting to please the masses. For a business to be able to directly gauge the response of their customers or users by the words they say helps to ensure that the decisions made are of sound mind. With the recent invention of Social Media, opinions and emotions are a daily discussion. Everyone has a voice online and doesn't always come with a rating. Yet, with the development of Sentiment Analysis models, the process of "Opinion Farming" can convert the opinions of the online world into direct numerical values to determine and represent those who are involved with said businesses [1].

Sentiment Analysis often relies on the use of Natural Language Processing (NLP) tools and methodologies to represent text phrases and sentences as numerical values that define the overall sentiment of said text. Considering the variable nature of language as a whole, it can be difficult to represent text as numerical values, thus additional processing is required to achieve this. Many different techniques of representing words in a way that can be interpreted by a model choose to represent these texts as vectors. This can be done through different techniques that either represent words based on their frequency or commonality between sentiments and then join them or by representing the phrases directly as vectors [4]. Vectors can be adapted into features to be learned by a Genetic Programming model to create a function that will best fit this vector. This will result in a sentiment value based on this grouping of generated features.

This is the aim of the project. To develop a Genetic Programming (GP) model, with the application of Symbolic Regression techniques to solve the NLP problem of Sentiment Analysis. Symbolic Regression is an important regression method related to the Evolutionary Computational field, Symbolic Regression allows a model to evolve programs represented by equations to fit the given data and output a numerical value. Symbolic Regression was chosen for this project due to the high interpretability of Symbolic Regression model outputs along with the usefulness of continuous results. Many models within the Sentiment Analysis space are classification models, this results in binary classifications of text and does not provide the depth of insight that the text itself represents. By using Symbolic Regression to derive a fit function, the representation of this text through sentiment can be enhanced and better understood.

GP was chosen for this project due to its efficiency and output design along with the applicable benefits of using it. For example, learning the output based on vectors would result in a classification model [5]. However, by classifying data limited to direct sentiment classifications, the potential accuracy of said opinion farming is lost. Thus, by generating a function, the output can be variable and can directly represent the sentiment of the piece in a continuous fashion that is easily interpretable and malleable for additional Machine Learning processes. To accurately evaluate the validity of this model, different fitness functions will be applied and their convergence curves will be evaluated. Then, the best-performing model will be compared to existing solutions within the space. The preprocessed data will be compared against a pre-trained Sentiment Analysis Convolutional Neural Network. A successful model will be defined as one that performs well compared to other models based on computational time and interpretability of response.

Considering the nature of the project, no further concerns need to be addressed on sustainability or environmental effects. The project is defined as entirely digital and will not leave any physical negative traces physically. As this model is developed entirely digitally and uses datasets ethically sourced, there are no intellectual property issues or environmental concerns birthed from the undertaking of this project.

II. Background Research

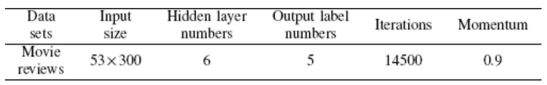
A. Literature Review

This section outlines solutions for Sentiment Analysis, pertaining to Genetic Programming and the methods used comparatively against the current solution defined. Also, other solutions will be investigated like CNNs or other Neural Networks to compare functionality and complexity. Defining the positives and negatives of previous solutions will help to further development in this current solution along with evaluating the overall fitness of this project in the NLP sphere.

1) Sentiment Analysis: Sentiment Analysis is an important problem relating to the field of Natural Language Processing. By being able to accurately gain insight into the sentiment of text, companies and businesses can better evaluate the reception of their goods and services without having to perform guided market analysis [24]. With the rise of social media in the communication space, opinions are more common than ever to see and hear [2]. However, not every comment or tweet comes with a score. This results in opinions being shared without a quantifiable result to garner from them for business insight. With the use of Sentiment Analysis, sufficient insight can be garnered by evaluating the very words used to form opinions online and around the internet as a whole. For example, businesses can use Sentiment Analysis on social media, targeting conversations about products launched to determine whether the product was successful and resonated with customers [25]. This dynamic gathering of information sees endless potential in an ever-evolving world of information. Recently, as deep learning models have become more prevalent in Machine Learning spaces, Sentiment Analysis and surrounding NLP tasks have been a common combination [26].

Convolutional Neural Networks aim to reduce the amount of parameters present in a standard Neural Network by convoluting the parameters to different layers. This helps more complex problems be learnable in reasonable amounts of time and to a higher degree of quality as large datasets are more viable and are optimal, especially for sentiment analysis. The quality of CNN-trained NLP Models can be very high due to the dense nature of learning. CNNs can recognise patterns between words and extract features from these connections between words, a very useful application for sentiment analysis due to the importance of semantic meaning. This can be done with little preprocessing compared to Genetic Programming and this makes for a less convoluted method of training a model.

One such study outlines the design and methodologies behind their implementation of a CNN for the use of Sentiment Analysis [11]. Firstly, this study retrieves over three million 300-dimensional vectors from a Google News dataset to train the CNN to recognise the trends of Word2Vec word embedding. With a limit of 53 words in a sentence, each instance has a max length of 53x300 values in a matrix to represent the sentence. This overall size of each instance is unsustainable for the scope and bounds of a Genetic Programming model. Thus, this type of preprocessing largely would be required to be avoided or implemented differently, one that is more reasonable for the evolution and development of a consistent but interpretable function. The overall parameters of the study's CNN are shown below.



Data sets	Input size	Hidden layer numbers	Output label numbers	Iterations	Momentum
Movie reviews	53×300	6	5	14500	0.9

Fig. 1: Study CNN Parameters

The study does not indicate an important preprocessing pipeline for ensuring that each word embedding is consistent semantically. Many of the aforementioned Genetic Programming models applied to this issue implemented extensive preprocessing pipelines to ensure that speech was consistent throughout different instances. This might have hindered the results garnered by this CNN and its subsequent tests as the variance in the text is large but can be reduced with the unification of language.

Another study outlining a CNN for Sentiment Analysis indicated a requirement to represent entire documents with a singular vector through the use of Doc2Vec [12]. Doc2Vec is an NLP library that contains a similar architecture to Word2Vec but aims to represent entire documents through a vector rather than singular words. The study outlined that Doc2Vec outperformed Word2Vec in many different datasets and is easier to implement due to the architecture. Doc2Vec uses separate word embeddings and creates a matrix of connections between documents, and thus each document can be represented as a single vector of fixed length.

2) Genetic Programming and Symbolic Regression: Genetic Programming, or GP, is an Evolutionary Computation method centred on the process of evolving programs through processes that improve over time. Genetic Programs mimic the evolutionary theory of the world by allowing programs to be evaluated on the fitness for their purpose, and subsequently either dominated by other programs or allowed to breed new ones to produce better-fit programs [27]. The superset of Evolutionary Computation has a wide variety of applications from data mining, business insights and search optimisation along with a wide variety of solution representations. These solutions can be represented as binary strings, arrays and even trees of functions. The purpose of models like GP is to evolve a fit program for the given requirements through continuous improvement and evaluation. The main processes that allow GP to function are;

- Reproduction: Copying of a program to the next generation.
- Crossover (Breed): A new program is created as a mix of two selected programs.
- Mutation: A new program is generated with a random change occurring to a part of an existing program.

- Evaluation: Programs are evaluated by a given method to determine the validity of the given problem.
- Selection: Programs are chosen for the aforementioned processes based on their evaluation quality.

These functions allow the model to continuously improve the programs generated and generate a fit-for-purpose program. These programs can be made up of a multitude of pieces. When GP generates a program, it pulls from a function set. This is a given set of operations allocated to the tree to be arranged and mutated throughout the training of the model. Each individual is evaluated using a fitness parameter for a set number of generations and then the best individual is chosen from the selection process. Design decisions related to the parameters of the model will be imperative to avoid when developing this project as maintaining the architectural indication of EC components will be better for the efficiency and interpretability of the model designed.

Symbolic Regression is a regression-based analysis method that aims to find an equation to represent given data. The goal of Symbolic Regression is to find a mathematical connection between data points that can represent a value dynamically. By representing the trends of the data using an equation rather than a series of choices, a quantifiable result can be interpreted from the analysis. This results in a more interpretable output from the analysis that can be better used for business insight and data collection.

Genetic Programming, particularly while implementing a Symbolic Regression analysis structure, specifically builds trees as the individuals [6]. The nodes of every tree are made up of operators and arguments, building an equation that can transform the given data into an interpretable result. An example of the conversion from the equation to a tree would look like this.

$$\frac{5}{2} - e^3$$

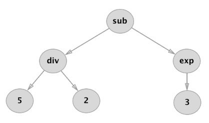
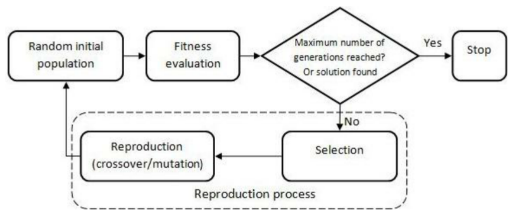


Fig. 2: 5/2 - e^3 represented as a GPSR Individual

The Genetic Programming model itself slowly evolves the trees through crossover of parents, mutations applied to nodes or even operators and fitness selections based on error calculation to optimise the trees to learn the trends of the data. The result of this process is a more variable and interpretable solution than other more conventional machine learning models.



Random initial population
Fitness evaluation
Maximum number of generations reached? Or solution found
Yes
Stop
No
Reproduction (crossover/mutation)
Selection
Reproduction process

Fig. 3: Flowchart of Genetic Programming Processes

3) Related Work on Genetic Programming for Sentiment Analysis: Applying Genetic Programming with Symbolic Regression to the problem of Sentiment Analysis can be difficult, thus it is imperative to research and evaluate existing examples of attempts made to create models for such a task. One study outlined its project to develop a Genetic Programming model for Sentiment Analysis [4]. Their model was trained on a dataset scraped from Twitter and contained 7,218 tweets with even spreads across the five classifications present in the dataset. Twitter and surrounding social medias are dense in their usage and can contain a large variety of differences in linguistic usage and even language itself. Twitter has over 150 languages represented online and expecting a preprocessing pipeline to represent all languages would be extensive [8]. Thus, this study normalises the data through four processes which can be summarised as;

- Error Correction: Removing misspellings/slang and special characters
- Part-Of-Speech Tagging: Giving specification of context with tags. Eg. in Spanish, ”Proud” is gendered as a masculine word, as such an ”M” tag is applied to the word.
- Negation Step: Negative connotation is applied to verbs or important adjectives based on the surrounding context.
- Filtering Step: The text is filtered to important contextual words.

This process helps to prepare the data for vectorisation while ensuring that the data is unified and will not have variance that detracts from the vectorisation strategies. This study uses Term Frequency - Inverse Document Frequency [9]. TF-IDF can be represented as such;

- TF = Frequency of word in sentence / Total words in sentence
- IDF = log(Total sentences in corpus / Total sentences with word in corpus)
- TF-IDF = TF * IDF

TF-IDF is used to determine the frequency of usage of a word within the entire corpus and how it relates to the semantics of the word. Naturally, words that are more commonly used regardless of sentiment, or stopwords, will have lower results from TF-IDF calculation. This ensures that important words are given a weight that defines how important the word is to the context of the sentence. This concept is used in tandem with a Latent Semantic Index (LSI), where instances are grouped by which contexts they occur. This preprocessing makes it so that each word has a direct importance, or weight attached as well as an intended context tag that allows for the expected vector to be generated with this information embedded within it. Considering the nature of online conversation and its complexity and lack of uniformity, the preprocessing steps applied in this study are imperative to the success of the model, as these processes are within every NLP problem [3].

The fitness function of a model is an important aspect, as it defines the quantifiable quality of each individual within the model. One study developed a model that used a fitness calculation named F1 Score to evaluate the individuals [7]. F1 Score can be used to represent the accuracy of estimates for binary classification by calculating the quality of guesses for each classification label separately. This allows for added insight into potential class biases and false positives that may occur during the classification process. To perform this evaluation, F1 values are calculated for each binary classification within the target variable and are compared against each other to determine which classification trend is more prevalent within the individual. F1 values measure the mean value between precision and recall for a binary classification. Precision measures how many of the predicted positive instances are correct, while recall assesses how many actual positive instances were correctly identified by the model. High precision means fewer false positives, which is important when false alarms are costly, whereas high recall indicates that most actual positives are captured, crucial when missing a positive is critical.

Considering that Sentiment Analysis can be treated as a binary classification from Positive to Negative, this is entirely viable. Precision being the amount of correct results and false positive results and recall being the ratio of correct results to the sum of correct results and false negative results. The function can be outlined like this;

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

The subsequent model produced with this fitness function and similar preprocessing steps to the previously mentioned study result in a competitive model comparatively to other implementations within the Sentiment Analysis research space. This is largely due to the high amount of preprocessing and domain knowledge applied to the datasets used along with the simplification of the instance's correct answer criteria. Performing only 3.6 F1 score below the SemEval 2014 methodology [10].

This model also performed a different evolution of typical Genetic Programming operators, rather than representing an equation using GP. This study represented the preprocessing pipeline using GP and allowed the model itself to control which data was fed into the classifier. This results in the model choosing operators to be applied to the instances of text.

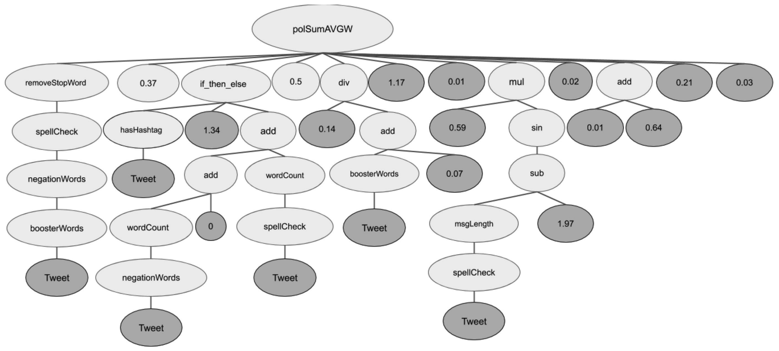


Fig. 4: Example of Preprocessing Node Representation from [7]

This example represented an evolved tree that would aim to find the positive or negative influence within a given text. Each branch of this tree would control a different preprocessing action. For example, the far left branch took in a tweet, split words into distinctions of importance along with their sentiment, then performed a spell check and then removed stop words. Other branches perform actions like checking for hashtags or adding together the results of other branches. This results in a tree that not only calculates the result of the instance but also controls how the text is represented. This gives the model more freedom in its application and helps with convergence on specificity.

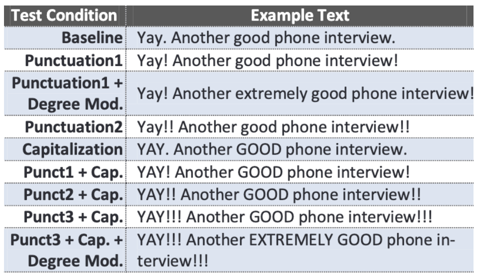
These studies and their methodologies directly influence the development and path of this project, as this project aims to increase the complexity of the model and yet further synthesise the problem with the design of a Symbolic Regression model. By designing and implementing a Genetic Programming model with Symbolic Regression the intent is to follow the design philosophy of these outlined studies while adapting the fitness and output to be variable in design and result in a regression problem rather than a classification problem. The intent behind this change is to improve the interpretability of the model's output while still intending to maintain the accuracy of the aforementioned examples. While a classification of sentiment is easy to understand, it loses a large amount of the important quantification that comes with a direct valence value, thus this design change necessitates the change from a classification model methodology to a regression model methodology. However, to further synthesise development understanding and ideas, other examples of models designed for Sentiment Analysis that are more detached from this project also require an evaluation.

4) Evaluation: Considering the breadth of tools and methodologies implemented in the different examples. A harmonious synthesis of the ideas presented would be a benefit to the design of this project. The use of vectorisation methods to represent the instances along with a lexicon for sentiment embedding into features would be an important step in preparing the data for use in training the GPSR model. An important distinction to make would be ensuring that the GP model itself is also tuned to allow for various nodes to be selected from, limiting this search space would remove the possible lowering of variance that necessitates specific nodes and operators. Doc2Vec as a tool is important for the application of vectorising text and can be used to help with the development of the preprocessing pipeline.

III. Tools and Methodologies

Considering the above insights gleaned from existing solutions and examples. This project intends to synthesise existing information to create a new solution in terms of a regression model. The planned tools to be implemented are vectorisation techniques like Doc2Vec for vector representation of instances and embedding of sentiment. Additional feature engineering will be handled by the VADER tool within the NLTK library and will be integrated with the vector representation to act as a base multiplier for features to harmoniously work together with the embeddings.

NLTK, or Natural Language Toolkit, is a library that is available for use in NLP problems in different languages and provides many important features that help with the development of NLP models. This includes preprocessing pipeline functions and feature extractors, however one package in the library that was used for feature engineering, VADER [13]. VADER is a Sentiment Analysis Lexicon that is integrated into NLTK's packages. This tool allows for words to be represented and mapped to a sentiment value with intensity scaling. This means that an entire sentence can be represented by sentiment values that are decided by a list of rules and their corresponding factors. For example, VADER includes rules that check for exclamation marks and increase the intensity of the model based on this. VADER also checks for capitalisation and applies the corresponding intensity increase. This rule-based design and change allows for instances to be differentiated based on their non-lexicographical information.



Test Condition	Example Text
Baseline	Yay. Another good phone interview.
Punctuation1	Yay! Another good phone interview!
Punctuation1 + Degree Mod.	Yay! Another extremely good phone interview!
Punctuation2	Yay!! Another good phone interview!!
Capitalization	YAY. Another GOOD phone interview.
Punct1 + Cap.	YAY! Another GOOD phone interview!
Punct2 + Cap.	YAY!! Another GOOD phone interview!!
Punct3 + Cap.	YAY!!! Another GOOD phone interview!!!
Punct3 + Cap. + Degree Mod.	YAY!!! Another EXTREMELY GOOD phone interview!!!

Fig. 5: Example of Rule-Based Lexicon Mapping

This tool could prove useful to this project's Genetic Programming model due to the simplicity of VADER's output. VADER outputs a sentiment value for positive, negative, neutral and compound values, these are direct quantifications of each side of the valence scale and will be imperative to the evolution of each individual within the corpus, The intention is to integrate this tool with the embedding systems to create a unique and efficient regression model. These values range from 0 to 1 and represent a score given to positive, negative, neutral and compound. With the compound value representing a normalised result of all three values ranging between -1 and 1. These values give specific scores to each potential classification of a given piece of text numerically and can be used as features for use within the model.

Important Python tools used in this model's design were DEAP and Pandas. DEAP is an evolutionary computation library for Python that is praised for its easy adoption and flexibility when developing different EC models [14]. This library allows for many different GP processes to be represented as tools in a toolbox, which can be called and accessed manually or can be controlled by algorithms designed for use with DEAP and its surrounding ecosystem. The simplicity of the model allows for quick development and understanding of the underlying systems of this GP model. The use of the Pandas library helps with the transmutation of data by providing easy-to-understand and efficient applications. Pandas is foundational for many different data and machine learning processes and their functionality [21]. This library is used to hold the data along with randomising and sampling the data during development for quick testing of the model and the pipeline.

Based on the investigations into fitness functions and the requirement of implementing a Regression model, steps were taken to represent different fitness functions like Mean Squared Error, Root Mean Squared Error and R^2. This will accurately represent the validity of each individual and help to regress the valence values to be accurate even in new instances. Other error functions will be tested alongside the most optimal and will represent the choices made during development. Other steps were taken to adapt and evolve the program by evaluating the validity of GP parameters and improving the model through these methods. The validity of the model will be quantified through graphing with the matplotlib library and will visually represent the convergence curve and thus the validity of the model.

IV. Method

A. Requirements

The requirements of this project are as such; produce a Genetic Programming model with Symbolic Regression for the task of Sentiment Analysis. This necessitates a preprocessing pipeline or control affirmed to the model itself to perform these operations. This model must use preprocessed features garnered from transmuting text data and be able to use it to define a valence-based result depending on the sentiment of the given text. This model will output a function that can best represent this data and provide outputs that are synonymous with the expected sentiment.

B. Design and Implementation

This model was designed to be easy to interpret and allow for ease of updating features. The main library used for this model's implementation was DEAP. A simple visualisation of the model can be described as such.

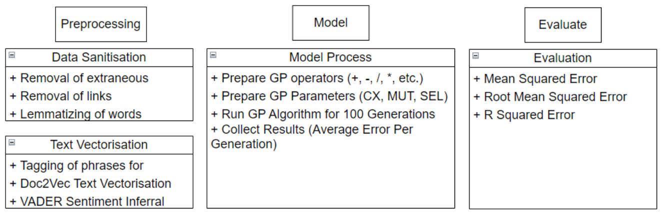

Preprocessing
Data Sanitisation
+ Removal of extraneous
+ Removal of links
+ Lemmatizing of words
Text Vectorisation
+ Tagging of phrases for
+ Doc2Vec Text Vectorisation
+ VADER Sentiment Inferral
Model
Model Process
+ Prepare GP operators (+, -, /, *, etc.)
+ Prepare GP Parameters (CX, MUT, SEL)
+ Run GP Algorithm for 100 Generations
+ Collect Results (Average Error Per Generation)
Evaluate
Evaluation
+ Mean Squared Error
+ Root Mean Squared Error
+ R Squared Error

Fig. 6: Diagram of Model Processes

This program was divided into three distinct files, which each have their own process and subset of features. This was done to improve the efficiency of development and separate sections of the codes for further development later in the project lifecycle where data would need to be collected from each section of the model pipeline.

1) Preprocessing: This first section of the model architecture represents the Preprocessing pipeline. This model necessitates the adaptation of text data into a vector of representative data that can accurately define the trends of the sentiment within the given corpus. The preprocessing pipeline was separated from the GP algorithm for ease of testing and to allow for scalable efficiency. Once the design of the preprocessing file is finalised, the methods can export the results once and skip having to run every time the GP model is required to train. Also, the models trained within this pipeline are important to the preparation of new data to be tested using the developed equation. By using the pre-fit Doc2Vec and VADER objects, new data can be represented as similar vectors to the training data, allowing the model to recognise and evaluate new data even when not used during training.

The processes that are applied during the preprocessing pipeline are varied and are a result of different datasets and requirements. With NLP problems, ensuring that data is synchronised across different instances to allow for trends to arise is imperative. In the current era of technology and communication, there is a greater amount of variance in the way people communicate and understand each other than ever before [15]. In the context of preprocessing, this necessitates uniformity in the data fed into the vectorisation methods. This is done through a pipeline of operations applied to each piece of text. This process can be listed as:

- Removal of Links (RegEx or Special Character Search)
- Removal of Online-Specific Data (Handles, Emojis)
- Removal of leftover Special Characters
- Lowering of Text Data and removal of extra spaces for Token handling
- Removal of Stop Words (and, the, is, etc.)
- Lemmatization of resulting words

This removal of links and online information was due to the initial dataset outlined for this project being one scraped from Twitter [16]. Online communication, as aforementioned, has many more inaccuracies and special characters within it such that preprocessing it to be as uniform as possible for vectorisation requires this pipeline. However, as the project progressed, the requirement for a deliberate Regression-based dataset arose. This resulted in a change to the Stanford Movie Review Database [17]. This dataset has 239,231 Phrases contained within it with attached sentiments. Many of these phrases have duplicates or substrings present within the dataset. This required large changes to the original design of the preprocessing pipeline. By applying the preprocessing pipeline that was outlined above, the amount of instances within the database decreases from 239,231 to 46,636 instances. This is largely due to the duplication of phrases and the subsetting of these instances. After removing stop words, removing extraneous characters and spaces, and lemmatizing the words in each piece of text. The duplicates are easy to filter using the Pandas Library, as it provides many different tools to easily transmute the entire data frame. The Stanford dataset was defined by a 1 to 25 scale. This could be classified into groupings of five but was more useful as a regression dataset. By averaging the three given sentiment classifications, we can use these continuous values to be learned and represented by a Regression Model. This necessitated improvement or adaptation of the Model's design later during the development of the project and the specific adaptation of the preprocessing pipeline.

The implementation of the preprocessing pipeline changed throughout development and as more details about the requirements of the data being created became clearer, changes were made to fit the results that were output. Originally, the model would receive vectors the size of 50 for each instance, this would represent the web of sentiment vectorised using the Doc2Vec model trained on the corpus before. These vectors would later be included with the implementation of the VADER sentiment analysis NLTK module. As mentioned in the literature review, this module uses a pre-made lexicon to reference words and extrapolate sentiment features for use in training. Lastly, the model would receive the actual class or true value in terms of the regression model. This data was hard to regress as the vectors did not directly represent the sentiment of the text but rather the connections between words and the instances true value.

When changing from a classification model to a regression model, the requirement for more interpretable data arose. Thus, a change to the preprocessing pipeline was made. A classifier would learn the trends of the dataset using the vectors from Doc2Vec and would give predictions on the same data about what it thinks the sentiment is. The classification would be split into groups of five. They can be represented like this:

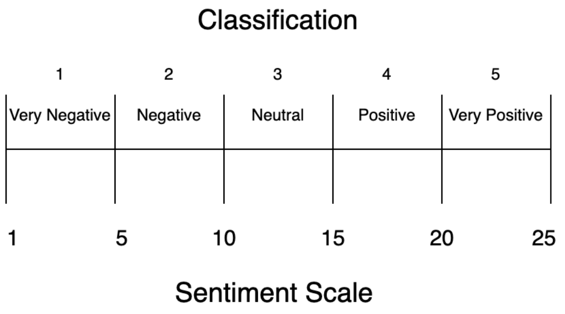


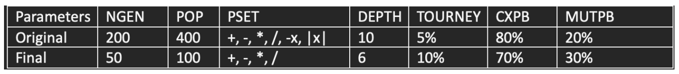
Fig. 7: Grouping of continuous values into classes

This results in a classifier making a prediction. This prediction is then adapted to fall within the continuous sentiment scale and fall in the middle of each classification's continuous value bounds. This result is then paired with the VADER lexicon information to inform the intent behind the prediction. This reduces the feature space by a large margin. Bringing the total features from 54 down to 5. This increases interpretability for the model and reduces variance in the output along with the requirements from the model itself to find the trends in the vectors as there is a smaller search space.

Algorithm 1 Preprocessing Pipeline Pseudocode

```
 1: Load Text and Sentiment into Frame
 2: Initialize VADER Sentiment Analyzer
 3: for each Row in Dataframe do
 4:     Clean text using CLEAN DATA
 5:     Save original for VADER lexicon
 6:     Save cleaned data for Doc2Vec Vectorisation
 7: end for
 8: Remove entries with empty content and duplicates
 9: Tag each sentence for Doc2Vec
10: Train Doc2Vec model and save model
11: for each Row in Dataframe do
12:     Infer vector using Doc2Vec model
13:     Compute sentiment scores using VADER on Non-Processed Text
14:     Concatenate vectors and sentiment scores into features
15: end for
16: Use Linear SVC to predict Classification and append Prediction as a feature
17: Remove Doc2Vec Vector from each instance
18: Export features to CSV
19: return features
```

2) GP for Generating Sentiment Analysis Model: The GP for Generating Sentiment Analysis Model throughout the project largely remains unchanged. Data is supplied by the preprocessing pipeline and trains a GP model on it. The model outlines the primitive set as the simple four operations; Plus, Minus, Multiply and Divide. Throughout the development of this project, the requirements of the parameters changed with the density of the data being input into the model. The original design of the preprocessing pipeline contributed to the original design of the GP Model having vastly different parameters to the final version



Parameters	NGEN	POP	PSET	DEPTH	TOURNEY	CXPB	MUTPB
Original	200	400	+, -, *, /, -x, |x|	10	5%	80%	20%
Final	50	100	+, -, *, /	6	10%	70%	30%

Fig. 8: Comparison of Parameters for Model

As the feature space dwindled due to the vectors moving to be used for classification estimates, the output to the model reduced significantly. This put less stress on the model to learn the vectors and instead did a large amount of work for the model so that the data could be saved once and the trends could be learned more easily during stochastic testing. The crossover and mutation operations stayed the same during development and would crossover two values at one point or replace a node with another in the primitive set. This lets the model evolve within confined bounds. The change in the percentages was due to the change in feature size as well. With fewer features, the likelihood of converging at local maximums was higher and the lack of density in the trees would allow for mutations to provide better results, thus the chance was increased from 20% to 30%. The parameters chosen were not specific to the problem but were chosen following default values for a successful Genetic Programming Model.

Another important part of the model is the two Terminal Set values set to 2.5 each. This is used in tandem with the VADER lexicon data, which are percentages of sentiment, to narrow down the specific continuous value within each classification indicated by the dataset. By doing this, the classification estimate can be used with this implementation to dynamically specify the sentiment within a classification bound and remove variance from the side of the model attempting to find evolved values that happen to fit all 46,636 instances. Giving two terminal values allows for more control and less variance of the output.

Processes like crossover, mutation and selection, which are typically done for Genetic Programming models are handled by DEAP for this project. This includes functions like the crossover of individuals, mutation and also the selection processes. The generations are also handled by DEAP as well with the "Algorithms" sub-package within DEAP handling the overall evolution of the model. However, this is the overall pseudocode for both the model sections handled manually and those done by the DEAP library and its subsidiaries;

Algorithm 2 GP for Sentiment Analysis

```
1:  Retrieve Data from Preprocessing Pipeline
2:  Create Toolbox for DEAP
3:  Set Parameters
4:  Create 2 Terminal Values (2.5 x 2)
5:  Initialise Crossover, Mutation, Select Paradigms
6:  for Generations in NGEN (50): do
7:      Generate Offspring
8:      Crossover (One Point Crossover)
9:      Mutation (Node Replacement)
10:     Calculate Fitness of Offspring:
11:     if $Chosen = MSE$ then
12:         $Fit = \frac{1}{n}\sum i = 1^n (y_i - \tilde{y_i})^2$
13:     else if $Chosen = RMSE$ then
14:         $Fit = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \tilde{x_i})^2}{N}}$
15:     else if $Chosen = R^2$ then
16:         $Fit = 1 - \frac{RSS}{TSS}$
17:     end if
18:     Select Best Offspring (Top 10%)
19: end for
```

3) Evaluation Methods: The original design for this model was to make a Symbolic Regression model using GP. This requires error calculation on the individuals rather than classification accuracy. The purpose of this is to allow for a continuous result from the Genetic Programming model that is better suited to the problem of Sentiment Analysis. Studies have shown that the accuracy for different training models based on classification and regression can vary, but in certain situations, regression models can be more accurate than their specified counterparts [18]. Also, regression model outputs allow for more interpretable results than classification models, especially when it comes to complex topics like Sentiment Analysis [19].

Three regression evaluation methods were chosen for this model to compare against in terms of the ability to help each individual converge and the overall fitness of the model itself. The first method is Mean Squared Error (MSE).

$$MSE = \frac{1}{n}\sum i = 1^n (y_i - \tilde{y_i})^2$$

The second is Root Mean Squared Error (RMSE).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \tilde{x_i})^2}{N}}$$

The third is the Coefficient of determination (R^2).

$$R^2 = 1 - \frac{RSS}{TSS}$$

These three functions control the validity of each individual and will help to converge the model on an evolved individual fit for purpose. MSE and RMSE will be important evaluation methods to determine the overall error of the regression model and its fitness to the data. R^2 defines the overall variance in the model comparatively to the data given [20]. These methods directly define the validity of the model for Sentiment Analysis.

Because of this, the results section will display the convergence curves of each fitness method and will define the best-fit fitness method for this model based on the steepness of the curve, as the values output are different but the shape of the curves will be compared on the same scale.

C. Model and Results

The GP model developed is a Symbolic Regression model. This model was trained using 46,366 instances of pre-processed phrases from a Stanford Sentiment Analysis dataset, This dataset was scraped from various movie review websites. Testing was done using a stochastic modelling process to evaluate variance within the model and its overall performance with differing seeds and generational changes. Each fitness function was tested 10 times and the convergence curve of the 10 tests is averaged to present the expected average result from training this model.

Also, to measure the validity of the model in the current NLP model climate, this model was compared through Mean Squared Error to a CNN trained and tested on the same data and split that this Genetic Programming model is trained on. This experiment will compare and contrast the effectiveness of the model in the NLP sphere and will provide insight into the model's overall fitness for its outlined purposes.

The model and the surrounding processes work as expected. The preprocessing methods adapt the text for use within the Doc2Vec model well and also reduce the feature size to only the applicable features. Also, the feature output is as expected with the VADER lexicographical data matching well to the true values in the data. However, many of the predictions made by the LinearSVC can be incorrect despite the large vector size given for each instance. The size of each instance's Doc2Vec Text Vectorisation Vector is 300 values. Despite this, many of the estimates can be wrong, even when reducing the search size from 1 - 25 to 1 - 5 by using the groupings listed above in the preprocessing section. This reduces the search size for the LinearSVC and allows for more accurate estimates compared to if the model was using the 1 - 25 whole numbers as class labels. Because other features to offset the possibility of incorrect estimates do not get generated alongside the estimate, the model cannot accurately recover from an incorrect estimate within the learning process. The model would have done better with more feature engineering on each instance or changing to a Strongly Typed GP model in which the model itself has control over the preprocessing pipeline, much like the study mentioned in the literature review section [7].

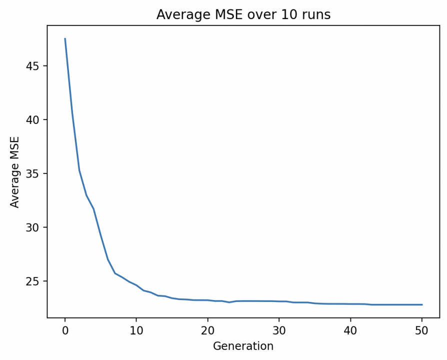


Fig. 9: Mean Squared Error Convergence Curve

1) Fitness Function Evaluation: The convergence curves are drawn by taking the best fitness at each generation for the ten runs completed and averaging the result. This gives a stochastic improvement of the model's performance across differing runs. Looking at the convergence curve for Mean Squared Error, the evaluation method shows a downward trend towards zero and improves over time. Mean Squared Error performs well in reducing the error but improvement slows at a given point due to the limitations of the data. The results suggest that the model has a difficult time pinpointing the specific class or value for the valence past using the estimate and VADER statistics. These errors could be offset by improving the estimates of the LinearSVC or providing more features to represent the specifics of the sentiment within each piece of text to allow the model to regress onto the true value

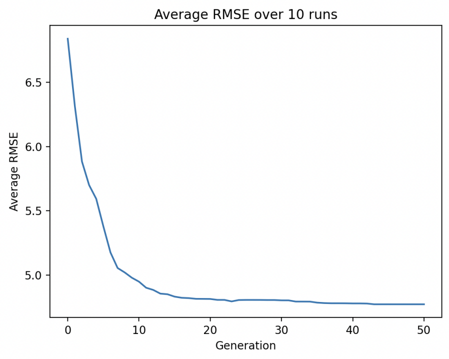


Fig. 10: Root Mean Squared Error Convergence Curve

Next, looking at the Root Mean Squared Error. This value gives a more proportional result to the actual scale used within the regression model. These two models have similar results and largely this is due to the same issues within the model itself. This insight into both error-based fitness functions suggests that further feature engineering would be required to allow for the model to recover in instances where the estimate is not correct. Or, engineering the vectors to hold the trends of the data better so that the LinearSVC model performs better and can more accurately pinpoint the class. This could be done with differing vector transformations or by reducing the search size for the LinearSVC to only two classes and leaving more of the feature engineering and trend learning to the Genetic Programming model. More feature engineering to specify the sentiment within a classification would allow for the regression model to take the improved estimates and use extra data to pinpoint the regression line.

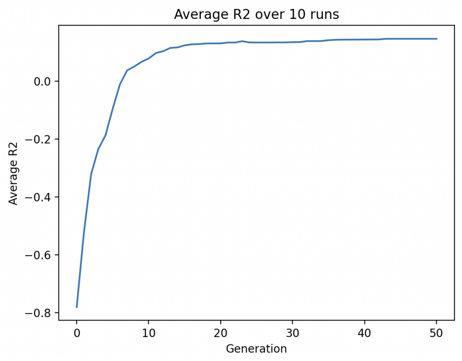


Fig. 11: R Squared Convergence Curve

Lastly, looking at the R^2 Results. This fitness function evaluates the total variance of the model and the amount of results that are fully covered by the model's outputs. We see an initial negative value before the model converges just over zero, at around 0.16. This means that the model covers around 16% of the total variance that exists within the dataset. This is in line with the error seen with MSE and RMSE as they fail to converge on the specific values and thus have high average errors.

Reviewing the trends for each fitness function type, we see similar trends for all three fitness function types. However, with the RMSE and MSE having similar trends despite different result bounds, we can interpret this as the better alternative to the lack of information provided by using R^2 due to the lack of variance covered by the model. Going forward with the model, the predominant regression evaluation method would be either MSE or RMSE due to their parallel results.

Fitness Function	MSE	RMSE	R^2
Average Final Result	22.796	4.774	0.145
Standard Deviation	0.881	0.091	0.033
Best Final Result	22.267	4.719	0.166

Overall results of Testing (3 s.f.)

The final results of the model are outlined in the table above. This covers the average final result from each run, the standard deviation between them, and the overall best run using each fitness function. The high variation is due to the quality of the classifications, and the accuracy of the lexicon mapping done by VADER. The model can find trends and interpret them, but cannot accurately pinpoint the sentiment for a given piece of text with the current preprocessing pipeline and model structure that exists within this build of the project.

2) Convolutional Neural Network Comparison: Comparing the results of this model to the existing examples within the space is still important. This allows for the model to competitively display against those that have been tested and proven in the space comparatively to this model. Notably, the use of CNN for classification problems with large vectors is commonplace [11]. This is because of the high vector space due to the embeddings of sentiment within each text requiring large connections to be made across vector groupings. This necessitates comparing the model's sufficiency against a CNN built for Sentiment Analysis and evaluating the Genetic Programming model on the same basis that the example CNN uses, which is classification. The model chosen to represent the current space of Sentiment Analysis models is a CNN using the Google News 300 Dataset for Word2Vec pre-trained embeddings [23]. This model uses a Jupyter Notebook to preprocess and create the Convolutional Net that represents the connections in the CNN. This model is later trained using a 90/10 split for train and test data [22].

To compare the validity of this model against the Genetic Programming model, the chosen error calculation will be the Mean Squared Error. This will give a synthesised result that directly quantifies the fitness for the purpose that the GP model possesses.

Model	Train MSE	Test MSE
GPSR	22.373	22.138
CNN	0.145	0.417

Comparison of Models in Mean Squared Error

Both models were tested using a 90 / 10 split for training and testing. This synthesises the results with each other to accurately gauge the fitness of each function for Sentiment Analysis. Both models used the preprocessed Stanford dataset created by the preprocessing pipeline, this synthesises the dataset used and eliminates any oversight related to differing qualities of datasets and rests the results solely on the model's sufficiency.

When evaluating the Genetic Programming model using Mean Squared Error, we see that the results do not yet fully meet the current climate of NLP models. The CNN that was evaluated ran for three epochs, running the dataset through the Convolutional Net three times over to learn and represent the trends of the data. The preprocessing of the data is done by the model itself and the vectors are learned from a pre-trained Word2Vec Model. This project's model trains the Doc2Vec vectoriser using the dataset and uses a LinearSVC to estimate the result, and that is used as a feature for regression. By obfuscating the trends of the data through features, important trends between words are lost that can be interpreted by a Convolutional Neural Network easily, but not by a Genetic Programming model without significant overhead being applied to the computational requirements of the model or the tree size. The CNN takes 5m 26s to run three epochs and the Genetic Programming model takes 45m and 13s to run 50 epochs. This difference in time scale results in the actual quality of learning being overall lower in the Genetic Programming model as it converged early into the training time. The CNN not only outperforms the current GP model but also manages to learn the trends of the vectors within a shorter time scale.

These results suggest that the Genetic Programming model designed is not yet at a point in which it can compete against others in the NLP space. After investigating the shortcomings of the model, the lack of diversity in features and in which data they originate lends to the results of the LinearSVC classification dictating the quality of the estimation. Without a correct estimation during preprocessing, many of the instances of data will not converge on their true value due to the lack of features to define this result within the output. Another shortcoming of the model is not using a pre-trained Doc2Vec model for vector inferral, this would've alleviated any misconceptions about the quality of the vectors being input into the LinearSVC, including the increased computational time required for the preprocessing pipeline. Further development of the ideas present in the current build of this model is necessitated to bring the model up to the quality of others within the NLP space.

3) GP Tree Analysis: To further evaluate the quality of the model, analysing the output and comparing it to the expected output would provide insight into the potential shortcomings of the model when it comes to the generation of the individuals.

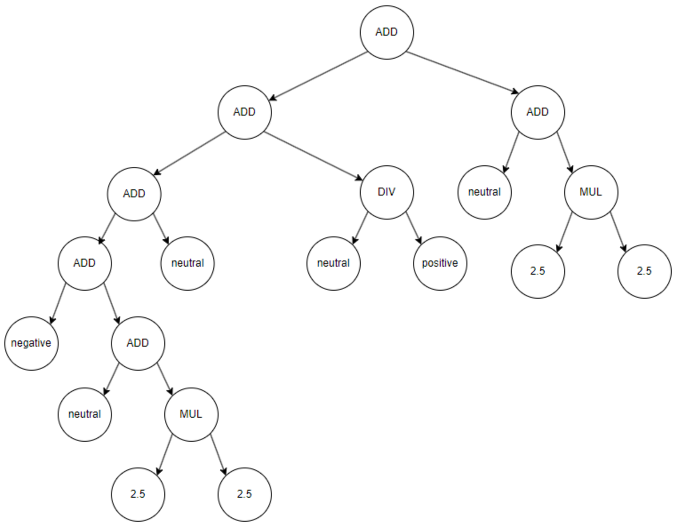


Fig. 12: Tree Generated using GP Model

This tree evolves using the model and takes in five features. The features are used to produce the result and output the best fit for the 46,636 instances of data. The features used in this tree are the VADER features, positive score, negative score and neutral. This tree does not use the sentiment estimate garnered from the Doc2Vec vectorisation estimation using LinearSVC. This is largely due to domination by more fit models even if the intended architecture is being reduced. Other features used are the terminal values outlined for pinpointing and the subset of the primitive set that is accessed does not contain the minus operation. This indicates that the values might overall be too low to reach values above a certain percent. Thus, to minimise the error, minus was removed to ensure that values on the higher end of the scale can be reached. This tree evolves differently than expected due to the removal of the sentiment estimate. The estimate generated using the LinearSVC is not used in this tree due to its inconsistency in providing an accurate and reliable result. This further strengthens the understanding that an important limitation of the model is the low consistency of the estimates from the Doc2Vec vectors. Thus, the GP program evolves to not use it and instead use the terminal set meant for specifying the continuous variable along with the VADER values. If the prediction quality or accuracy was improved or validated, this could change the tree structure generated by the GP algorithm and could improve performance significantly as an important feature is not used within this example of an evolved tree.

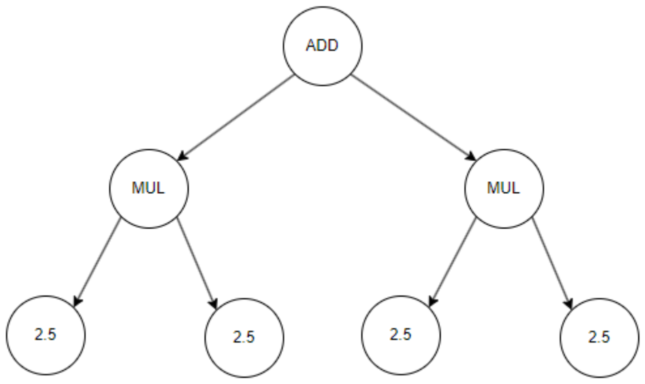


Fig. 13: Second Tree example generated using GP Model

This second example performs worse than the average mean Squared Error outlined in the stochastic testing but still becomes the best individual for the seed tested. This result shows that the features generated are not reliable enough for reducing the error, and when the model generates trees that reduce the error, regardless of the design or method, they dominate the others that use the features given, slowly until the unreliable features are removed. This means that the model has the potential to entirely remove the preprocessing pipeline implementation into the model. This goes against the intended design for the model and indicates that further work needs to be done to ensure that the features generated during preprocessing are reliable and consistent. This type of tree dominating the model could be mitigated by improving the quality of the estimates, or by engineering more features that alleviate the potential inaccuracies of the estimates and lexicon scores. Another potential way to mitigate the reduction of feature use would be to allow the model more options to transmute the data. This could be improved by adding more custom operations to the primitive set that allow the model to take a generalisation of the estimates through log functions or averaging features if more are added. Improving the reliability of these features in tandem with expanding the primitive set, would ensure that these features are used in trees and would improve the efficiency of the model by ensuring that the model is evolving quality programs along with allowing more options to the model to improve and evolve.

V. Conclusions and Future Work

Based on the conclusions collected from the investigations done during the development of this project, the Genetic Programming with Symbolic Regression model developed is currently not up to the same standards as others in the Natural Language Processing model space. The model is unable to pinpoint the valence of a given piece of text when attempting to regress an individual onto the trends of the data. However, the preprocessing pipeline results are easy to understand and the function outputs are interpretable due to the low feature space and small tree size.

The design decisions outlined in this report are synthesised with the findings in the literature review section along with understanding reviewed in similar papers. The model was designed to be simple to develop and produce interpretable results from the preprocessing pipeline for the ease of learning by using Genetic Programming. The efficiency of the model could be improved by lowering the training space or by improving the feature space to allow for better training quality. This would allow the model to perform better and alleviate the overhead allocated to the training time.

A. Limitations

An important limitation of the current model is the development process necessitating the use of the DEAP library. The library used is designed to be easy to use and understand while providing a variety of quality-of-life features for the use of EC computation. Also, the algorithms applied limit the data transmutation allowed during training without significantly changing the design of the model itself. Another important limitation of the model is the preprocessing pipeline. This pipeline uses a classification machine from Scikit-Learn to estimate the class of the instance and then append an estimate as a feature to the instance by using vectors generated using a CBOW model known as Doc2Vec. This could be improved significantly through various methods like using a pre-trained Doc2Vec model rather than training it on the dataset itself, or by improving these vectors before estimating on them, this could increase the quality of the estimates. This could also be alleviated by engineering more features relating to the instances of text that do not relate to the estimate.

To further improve this model, important steps would need to be taken to either improve the feature accuracy in terms of the classification estimate or engineer more features to alleviate the possible misclassification of a given instance. Improvements to the preprocessing pipeline would be required to ensure that the vectors being input into the LinearSVC are better representative of the sentiment in the text, or by using this pipeline to create more features through varying methods. Improvements could also be made by changing the structure of the model. By harmonising the model with the design in other studies referenced in this report, the model design could incorporate the preprocessing pipeline in the creation of the tree itself [7]. This model change would be significant but could improve the quality of the model significantly and allow for a more varied result that is easily interpretable and improvable with new functions and features being controlled by the Genetic Programming model itself. These changes could help the model realise its potential within the Natural Language Processing space.

REFERENCES

[1] M. Wankhade, A. C. S. Rao, and C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges," Artificial Intelligence Review, vol. 55, no. 55, Feb. 2022, doi: https://doi.org/10.1007/s10462-022-10144-1.

[2] Y. Mao, Q. Liu, and Y. Zhang, "Sentiment analysis methods, applications, and challenges: A systematic literature review," Journal of King Saud University. Computer and information sciences/Magalat gam'at al-malik Saud : ulm al-ḥasib wa al-ma'lumat, vol. 36, no. 4, pp. 102048–102048, Apr. 2024, doi: https://doi.org/10.1016/j.jksuci.2024.102048.

[3] E. Haddi, X. Liu, and Y. Shi, "The Role of Text Pre-processing in Sentiment Analysis," Procedia Computer Science, vol. 17, pp. 26–32, 2013, doi: https://doi.org/10.1016/j.procs.2013.05.005.

[4] M. Graff, Eric Sadit Tellez, Hugo Jair Escalante, and Sabino Miranda-Jiménez, "Semantic Genetic Programming for Sentiment Analysis," pp. 43–65, Jan. 2017, doi: https://doi.org/10.1007/978-3-319-44003-3_2.

[5] Nurulhuda Zainuddin and A. Selamat, "Sentiment analysis using Support Vector Machine," ResearchGate, Sep. 2014. https://www.researchgate.net/publication/286583940_Sentiment_analysis _using_Support_Vector_Machine (accessed Sep. 29, 2024).

[6] D. Angelis, Filippos Sofos, and T. E. Karakasidis, "Artificial Intelligence in Physical Sciences: Symbolic Regression Trends and Perspectives," vol. 30, no. 6, pp. 3845–3865, Apr. 2023, doi: https://doi.org/10.1007/s11831-023-09922-z.

[7] A. B. Junior, N. F. F. da Silva, T. C. Rosa, and C. G. C. Junior, "Sentiment analysis with genetic programming," Information Sciences, vol. 562, pp. 116–135, Jul. 2021, doi: https://doi.org/10.1016/j.ins.2021.01.025.

[8] T. Alshaabi et al., "The growing amplification of social media: measuring temporal and social contagion dynamics for over 150 languages on Twitter for 2009–2020," EPJ Data Science, vol. 10, no. 1, Mar. 2021, doi: https://doi.org/10.1140/epjds/s13688-021-00271-0.

[9] W. Uther et al., "TF–IDF," Encyclopedia of Machine Learning, pp. 986–987, 2011, doi: https://doi.org/10.1007/978-0-387-30164-8_832.

[10] X. Zhu, S. Kiritchenko, and S. M. Mohammad, "NRC-Canada-2014: Recent Improvements in the Sentiment Analysis of Tweets," Jan. 2014, doi: https://doi.org/10.3115/v1/s14-2077.

[11] X. Ouyang, P. Zhou, C. H. Li and L. Liu, "Sentiment Analysis Using Convolutional Neural Network," 2015 IEEE International Conference on Computer and Information Technology;Liverpool, UK, 2015, pp. 2359-2364, doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.349.

[12] M. Rhanoui, M. Mikram, S. Yousfi, and S. Barzali, "A CNN-BiLSTM Model for Document-Level Sentiment Analysis," Machine Learning and Knowledge Extraction, vol. 1, no. 3, pp. 832–847, Jul. 2019, doi: https://doi.org/10.3390/make1030048.

[13] C. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text," Proceedings of the International AAAI Conference on Web and Social Media, vol. 8, no. 1, pp. 216–225, May 2014, Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14550/14399

[14] J. Kim and S. Yoo, "Software review: DEAP (Distributed Evolutionary Algorithm in Python) library," Genetic Programming and Evolvable Machines, vol. 20, no. 1, pp. 139–142, Nov. 2018, doi: https://doi.org/10.1007/s10710-018-9341-4.

[15] D. Salih and A. - Khazaali, "Linguistic Strategy in Online Digital Communication: A Pragmatic Study," Journal of Media Culture and Communication, no. 43, pp. 10–21, May 2024, doi: https://doi.org/10.55529/jmcc.43.10.21.

[16] "Twitter Sentiment Analysis," www.kaggle.com. https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis

[17] "Deeply Moving: Deep Learning for Sentiment Analysis," Deeply Moving: Deep Learning for Sentiment Analysis. https://nlp.stanford.edu/sentiment/index.html

[18] Siirtola and J. Röning, "Comparison of Regression and Classification Models for User-Independent and Personal Stress Detection," Sensors, vol. 20, no. 16, p. 4402, Aug. 2020, doi: https://doi.org/10.3390/s20164402.

[19] J. Martinez-Gil and J. M. Chaves-Gonzalez, "A novel method based on symbolic regression for interpretable semantic similarity measurement," Expert Systems with Applications, vol. 160, p. 113663, Dec. 2020, doi: https://doi.org/10.1016/j.eswa.2020.113663.

[20] Induraj, "Which Metrics in Regression matter the most? MSE—RMSE—MAE—R2—Adj R2- Advantages/Disadvantages," Medium, Feb. 22, 2023. https://induraj2020.medium.com/which-metrics-in-regression-matter-the-most-mse-rmse-mae-r2-adj-r2-advantages-disadvantages-55740cb873ec

[21] W. Mckinney, "pandas: a Foundational Python Library for Data Analysis and Statistics," ResearchGate, 2011. https://www.researchgate.net/publication/265194455_pandas_a_Foundational _Python_Library_for_Data_Analysis_and_Statistics

[22] saadarshad102, "GitHub - saadarshad102/Sentiment-Analysis-CNN: Sentiment Analysis using Convolution Neural Networks(CNN) and Google News Word2Vec," GitHub, 2019. https://github.com/saadarshad102/Sentiment-Analysis-CNN (accessed Oct. 10, 2024).

[23] KA-KA-shi, "GoogleNews-vectors-negative300 (word2vec)," Kaggle.com, 2021. https://www.kaggle.com/datasets/adarshsng/googlenewsvectors (accessed Oct. 10, 2024).

[24] K. L. Tan, C. P. Lee, and K. M. Lim, "A Survey of Sentiment Analysis: Approaches, Datasets, and Future Research," Applied Sciences, vol. 13, no. 7, p. 4550, Apr. 2023, doi: https://doi.org/10.3390/app13074550.

[25] M. Birjali, M. Kasri, and A. Beni-Hssane, "A comprehensive survey on sentiment analysis: Approaches, challenges and trends," Knowledge-Based Systems, vol. 226, no. 1, p. 107134, Aug. 2021, doi: https://doi.org/10.1016/j.knosys.2021.107134.

[26] A. Yadav and D. K. Vishwakarma, "Sentiment analysis using deep learning architectures: a review," Artificial Intelligence Review, Dec. 2019, doi: https://doi.org/10.1007/s10462-019-09794-5.

[27] J. R. Koza and R. Poli, "Genetic Programming," Search Methodologies, pp. 127–164, 2021, doi: https://doi.org/10.1007/0-387-28356-0_5.