BT4222 Mining Web Data for Business Insights
AY 2021/22 Semester 1
Group Project

Suicidal Text Detection

in Social Media Posts

Prepared by: Group 8

Aeron Sim Shih Win	A0180347E	e0273924@u.nus.edu
Goh Jia Yi	A0185610J	gohjiayi@u.nus.edu
Lim Zi Hui	A0187781M	limzihui@u.nus.edu
Lin Xiao	A0202619L	e0418026@u.nus.edu
Quek Yi Zhen	A0190357B	e0325941@u.nus.edu

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Problem

According to the Singapore Mental Health Study conducted by the Institute of Mental Health (IMH), depression is the most common mental disorder — with 1 in 16 people in Singapore having the condition at least once in their lifetime (Choo, 2018). In particular, 18% of youths in Singapore, between the ages of 10 to 29, are estimated to suffer from depression (Promises Editorial Team, 2020).

Youths are more prone to develop depression due to chemical changes in the brain arising from puberty or unhealthy levels of stress from societal and academic pressures. The COVID-19 pandemic has further exacerbated this problem with pandemic restriction measures in place leading to school closure. In turn, the lack of human interaction increases the level of insolation felt by these individuals. For some youths, long periods of time spent at home with their families might cause unhappiness and frustration due to the lack of privacy. With these factors in play, the rate of depression among youths is expected to increase. This becomes a concerning social issue as the Singapore Mental Health Study has revealed that depression is strongly associated with suicidal ideation (Ganesh et al., 2016).

According to the Samaritans of Singapore (SOS), suicide remains the leading cause of death among youths in Singapore. In 2020, the number of suicide cases in Singapore has reached its highest in 8 years amid the pandemic (Tham, 2021). Thus, timely detection for the signs of suicidal intent amongst affected youths is crucial for intervention and treatment before the situation escalates into an irreversible stage.

1.2 Background

According to the Singapore National Youth Survey, more than 98% of youths communicate with their friends and families online (National Youth Council, 2019). As avid users of social media and digital technologies, youths are more likely to communicate about suicidal thoughts online. Thus, there is a pressing need to develop efficient big data systems to extract suicidal intent from social media data through advanced Natural Language Processing (NLP) techniques.

1.3 Objective

This project aims to build a predictive model to detect suicidal intent in social media posts. To further highlight the applicability of our suicide detection model, we aim to integrate it into a functional chatbot designed to build intimate conversations with youths and direct them to professional resources when signs of distress are identified.

2. Data Collection and Wrangling

2.1 Data Collection

The *Suicide and Depression Detection dataset*[1] used for this project was obtained from Kaggle, which consists of posts from the social media platform Reddit.

	text	class
0	Ex Wife Threatening SuicideRecently I left my ...	suicide
1	Am I weird I don't get affected by compliments...	non-suicide
2	Finally 2020 is almost over... So I can never ...	non-suicide
3	i need helpjust help me im crying so hard	suicide
4	I’m so lostHello, my name is Adam (16) and I’v...	suicide

Figure 1. Sample Rows of Original Dataset

The dataset consists of 2 columns as seen in Figure 1 above, where “text” indicates the post content and “class” indicates the label of the posts. A total of 232,074 posts were scraped from 2 subreddits, namely *SuicideWatch* and *teenagers* to build this dataset. Posts from *SuicideWatch* were labelled as suicidal, while the posts collected from *teenagers* were labelled as non-suicidal.

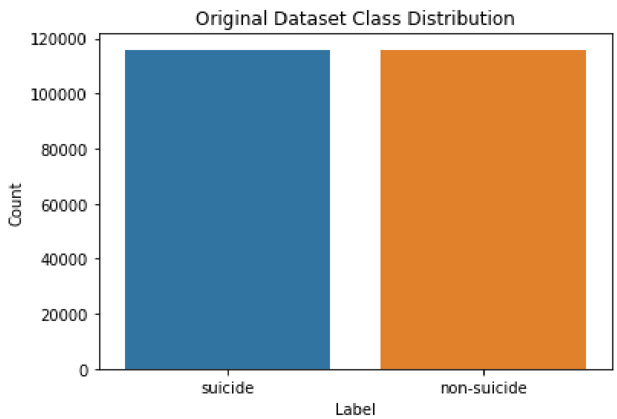


Figure 2. Original Dataset Class Distribution

The classes are distributed equally as seen in Figure 2 above where there are 116,037 rows, representing 50% of the dataset, within each class.

[1] Dataset accessible at https://www.kaggle.com/nikhileswarkomati/suicide-watch.

2.2 Text Preprocessing

The text data requires preprocessing to prepare the data into suitable formats for the subsequent model building. Social media data tends to be more unstructured and require more customised preprocessing and cleaning processes. Thus, our data was cleaned with the following steps in the sequence seen in Figure 3 below.

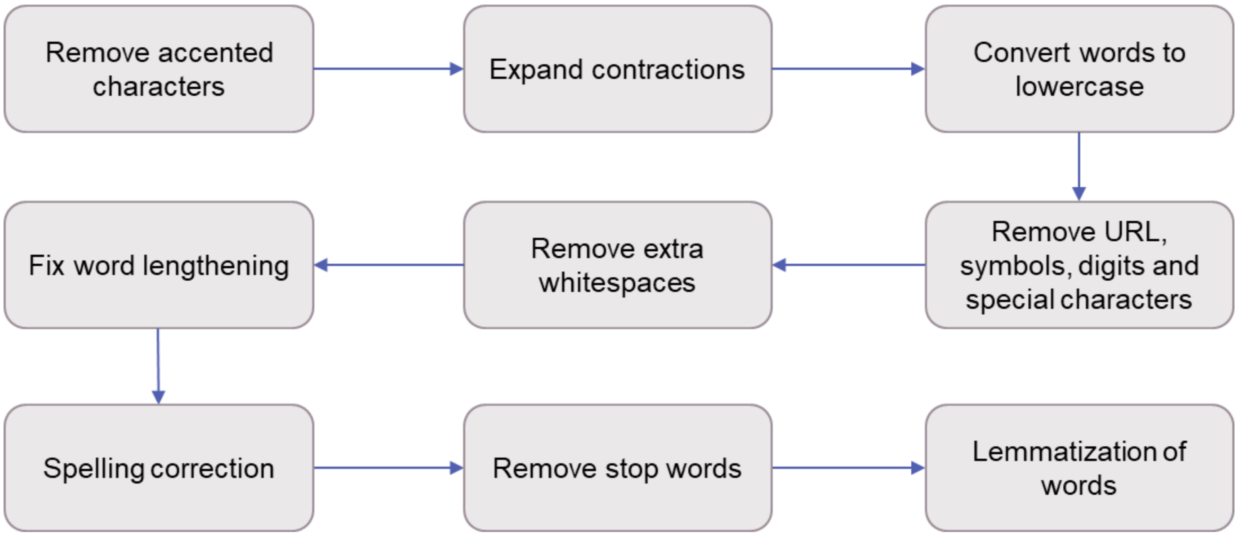


Figure 3. Data Preprocessing Steps

Remove accented characters. Words such as “cafe” and “café” have the same meaning. Hence, accented characters are removed to represent similar terms as the same word. This step also serves as a dimensionality reduction step to reduce the dataset vocabulary size. This is ideal as we wish to avoid high-dimensional vectors, which may increase the computational complexity for the model training process.

Expand contractions. Contractions are shortened forms of words where certain letters are removed. For example, the word “I’ve” is a contracted form of “I have”. As part of text standardisation, contractions are expanded to their original texts. This is also a useful step for dimensionality reduction before generating word vectors.

Convert to lowercase. Words with different cases such as “Boy” and “boy” have the same meaning hence they should be represented as the same word. By converting all words into lowercase, this also helps to reduces the dimensionality of the data.

Remove URLs, symbols, digits, special characters, and extra whitespaces. These characters do not contain important information for our use case and are removed since they are irrelevant for our model. The removal of these elements is carried out from a left-to-right order.

Fix word lengthening. Word lengthening is a result of characters being wrongly repeated. For example, “goooood” is a lengthened version of “good”. Since English words have a maximum of two repeated characters, other unnecessary characters need to be removed to prevent misleading information from being incorporated into our prediction model.

Spelling correction. The *Symspell* algorithm is used for spelling correction. It is based on the *Symmetric Delete Spelling Correction* algorithm, which performs faster than conventional spelling packages. In particular, *Symspell* also supports compound-aware automatic spelling correction of multi-word input strings (Garbe, 2017). This helps to correct two words that were combined due to the omission of whitespace. For example, “helpjust” is rectified to “help just” with a space character to indicate their word boundaries.

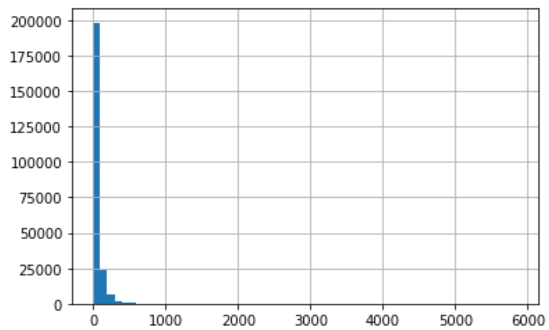
Remove stopwords. Stopwords are words that are commonly occurring yet do not carry importance or meaning. A few of such examples include “a”, “that”, and “the”. Although “no” and “not” are found in the list of commonly used stopwords, they have been retained as they imply a negative intonation, which may be useful in the context of classifying suicidal posts.

Lemmatization. Lemmatization is the conversion of inflected words into their root form. This is done to map different forms of a word to a root word since they represent the same meaning. For example, the terms “eat”, “eating”, and “eaten” contain the same meaning and could be represented with the same term when being fed into models. In contrast to stemming, lemmatization conducts a morphological analysis of the words, considering the contextual meaning behind the word. An example would be the lemma of “better” is “good”, but stemming would not be able to recognise this, therefore lemmatization is chosen over stemming.

2.3 Data Cleaning

Remove Irrelevant Words. We then conducted some initial data exploration on the number of occurrences for each word in the preprocessed posts. Among the top 30 most frequently occurring words, we noticed an anomalous word, "filler", which had 55,442 occurrences. Upon further exploration, we have concluded that this token does not have any meaning and is most likely a noise captured during the data collection stage. Hence, we proceeded to remove this word from all posts.

Remove Empty Rows after Preprocessing. After text preprocessing and removal of irrelevant words, there were approximately 60 empty text fields that do not contain any words. These corresponding rows were dropped.



count	232009.000000
mean	53.143029
std	89.614739
min	1.000000
25%	12.000000
50%	25.000000
75%	62.000000
max	5850.000000

Figure 4. Histogram and Distribution of Word Count in Posts

Remove Outlier Rows with High Word Count. We went on to obtain the word count of the preprocessed posts. From Figure 4 above, we observed outliers in the word count with extremely large numbers. The maximum of 5,850 words deviates greatly from the 75th percentile of 62 words. To optimise for model training in later stages, we have decided to use the 75th percentile as a threshold for word count where processed posts with more than 62 words are removed. Subsetting the dataset will allow us to focus on model training on shorter posts and reduce training time.

Final Dataset

	class	text	cleaned_text
0	suicide	Ex Wife Threatening SuicideRecently I left my ...	sex wife threaten suicide recently leave wife ...
1	non-suicide	Am I weird I don't get affected by compliments...	weird not affect compliment come know real lif...
2	non-suicide	Finally 2020 is almost over... So I can never ...	finally hear bad year swear fucking god annoying
3	suicide	i need helpjust help me im crying so hard	need help just help cry hard
4	suicide	It ends tonight.I can’t do it anymore. \nI quit.	end tonight not anymore quit

Figure 5. Sample Rows of Cleaned Dataset

The final cleaned dataset consists of 174,436 rows as seen in Figure 5 above where the “cleaned_text” column contains the processed text fields.

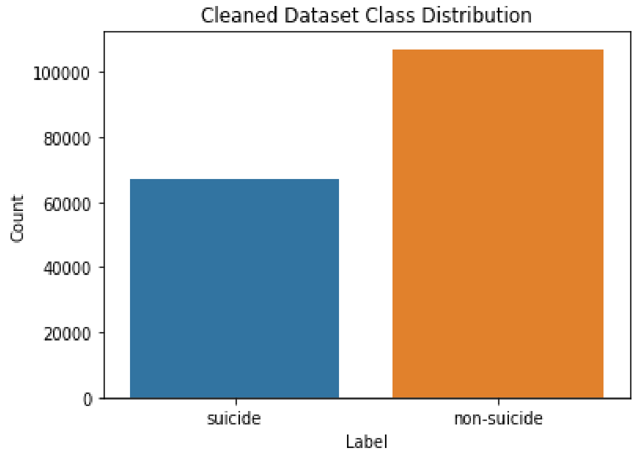


Figure 6. Cleaned Dataset Class Distribution

The class distribution of the cleaned dataset is shown in Figure 6 above. Compared to the original dataset class distribution (refer to Figure 2 above), the 5:5 ratio between suicidal and non-suicidal text has changed to an approximately 4:6 ratio. From this observation, we can infer that more suicidal posts had higher word counts which were removed as part of data cleaning. Although there is a slight imbalance in the class distribution, it can still be regarded as a normal classification problem and model performance will not be affected (Brownlee, 2020).

3. Data Exploration

We first split the dataset into train, test, and validation sets[2] with a ratio of 8:1:1. After which, we conducted data exploration on the train set and further zoom into each class.

Word Cloud

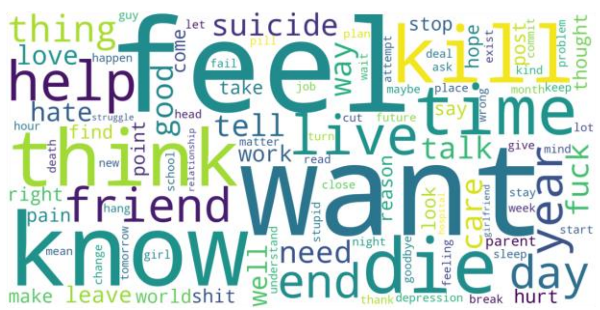
Suicidal Text	Non-suicidal Text
	

Figure 7. Word Cloud for Suicidal and Non-suicidal Text

We first explored the top 100 most frequent words and presented them in the form of word clouds in Figure 7 above. In word clouds for both suicidal and non-suicidal texts, we observed commonly used verbs such as “feel”, “know”, and “want”. Interestingly, suicidal texts are filled with negative words like “kill”, “die”, “hate”, “end”, and “help”. In contrast, non-suicidal texts were packed with more neutral and positive words like “good”, “people”, “friend”, “girl”, and “guy”. This observation suggests that suicidal texts tend to express more negative sentiments than non-suicidal texts.

Average Text Length Distribution

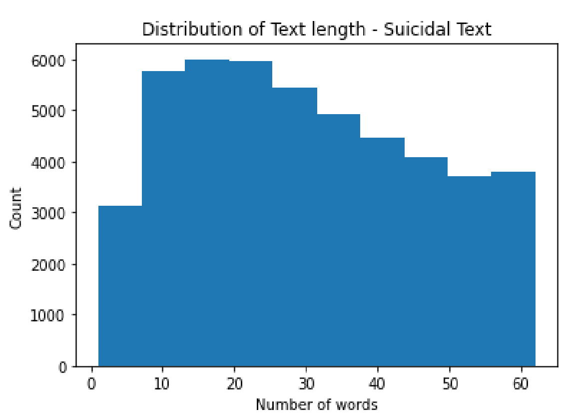
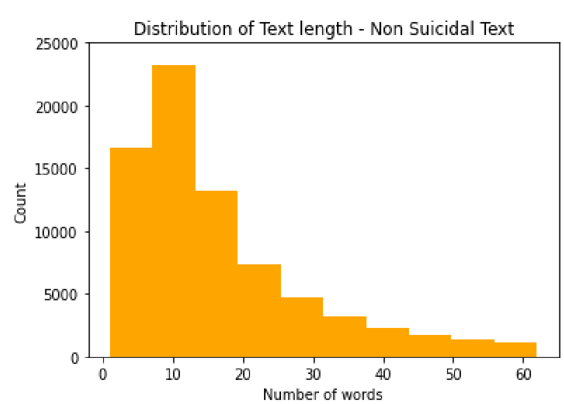
Suicidal Text	Non-suicidal Text
	

Figure 8. Distribution of Text Length for Suicidal and Non-suicidal Text

[2] Train dataset consists of 139,548 rows, while test and validation datasets consist of 17,444 rows each.

We explored the distribution of the average text length of posts in the form of histograms in Figure 8 above. Suicidal texts have a more uniform distribution as compared to non-suicidal texts, which have a right-skewed distribution. This means that non-suicidal texts are generally shorter than suicidal texts. From this observation, we can infer that individuals with suicidal thoughts tend to have more feelings to convey and write longer posts, which aligns with our initial inference.

Polarity Score Distribution

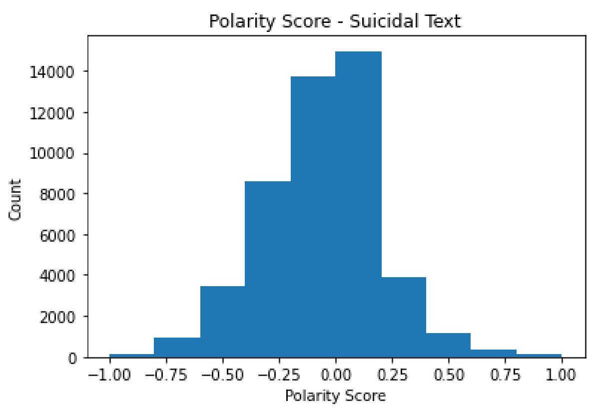
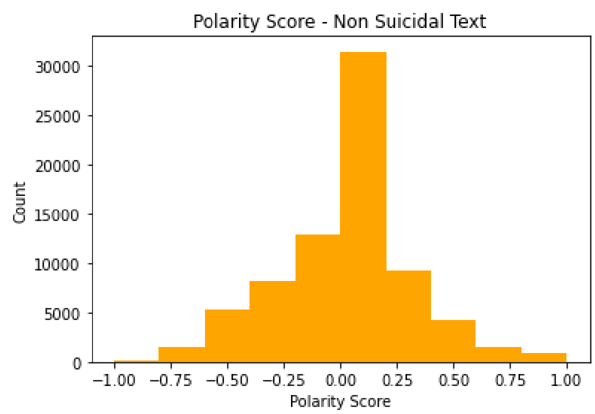
Suicidal Text	Non-suicidal Text
	

Figure 9. Polarity Score for Suicidal and Non-suicidal Text

We explored the polarity score of texts which ranges from -1 to 1, where -1 represents negative, 0 stands for neutral, and +1 represents positive sentiment. The distributions of polarity scores obtained are seen in the histograms in Figure 9 above.

In general, both distributions follow a bell-shaped distribution. Suicidal texts have more data points on the left side of the distribution, representing more negative sentiment posts. On the other hand, most non-suicidal texts are classified as neutral. This observation is consistent with our initial intuition that suicidal text has a more negative sentiment.

Most Frequent Bigrams

Suicidal Text		Non-suicidal Text	
Bigram	**Count**	**Bigram**	**Count**
('feel', 'like')	10405	('feel', 'like')	3291
('want', 'die')	5398	('year', 'old')	1299
('want', 'kill')	2365	('want', 'talk')	1248
('want', 'end')	1872	('sub', 'edit')	1198
('commit', 'suicide')	1653	('min', 'craft')	1144
('want', 'live')	1636	('need', 'help')	1121
('suicidal', 'thought')	1560	('good', 'friend')	939
('end', 'life')	1392	('look', 'like')	865
('need', 'help')	1381	('want', 'know')	844
('year', 'old')	1280	('high', 'school')	692
('go', 'kill')	1175	('want', 'chat')	654
('good', 'friend')	1136	('amp', 'amp')	642
('get', 'bad')	1115	('discord', 'server')	628
('die', 'want')	1071	('girl', 'like')	609
('know', 'anymore')	996	('real', 'life')	592
('anymore', 'want')	971	('video', 'game')	564
('need', 'talk')	939	('want', 'play')	551
('feel', 'bad')	928	('like', 'know')	508
('know', 'want')	920	('year', 'ago')	491
('want', 'talk')	886	('feel', 'bad')	472

Figure 10. Bigram for Suicidal and Non-suicidal Text

Bigrams represents a pair of consecutive words in a text. We explore the most frequently occurred bigrams in Figure 10 above, which can provide insights into the dataset. Most frequent suicidal bigrams include examples such as “want die”, “want kill”, and “commit suicide”, while non-suicidal bigrams include “year old” and “good friend”. This suggests that individuals writing suicidal texts are often experiencing extreme negative emotions, highlighting the urgent need to receive prompt help. As the bigrams were generated after text preprocessing, the bigrams generated from the original sequence of words are not our focus.

4. Representation Learning

In the context of NLP, representation learning refers to a set of techniques that transform raw textual data into a computationally efficient representation that is useful for machine learning tasks. As natural language texts are typically unstructured, with multiple granularities, tasks, and domains, it makes it challenging for NLP to achieve satisfactory performance. Therefore, word embeddings are commonly used to represent words in the form of a dense vector. They allow machine learning classifiers to capture semantic and syntactic information of words and identify other words with similar contexts. Models such as Word2Vec, GloVe, and fastText are commonly used for deep learning model initialisation due to higher effectiveness as compared to a random initialisation.

Without loading pre-trained word embeddings into the embedding layer, our model will have to learn the word embeddings from scratch during training which could be challenging due to the following reasons. Firstly, our training data could be sparse with a significant number of rare words. Embeddings learnt might not accurately represent these words due to the lack of information. Secondly, by learning the embeddings from scratch, our training process will be significantly longer due to the large number of trainable parameters.

Figure 11. Representations Built

In this section, we aim to derive representations using the Word2Vec and GloVe algorithms. Custom Word2Vec embeddings will be pre-trained using the training dataset while a readily available GloVe embeddings will be used (as seen in Figure 11 above).

4.1 Word2Vec

Word2Vec embeddings can be used to provide a dense vector representation of words and capture information about their meaning. Its effectiveness stems from grouping together vectors of similar words and making estimates about the meaning of the word based on their occurrences in the text. The estimates provide word associations with other words in the corpus (Vatsal, 2021).

Algorithm

Specifically, Word2Vec is a prediction-based method built on neural networks and consists of 2 algorithms: (1) *continuous bag-of-words (CBOW)* and *skip-gram (SG)*. CBOW tries to predict a target word based on the list of context words, while SG takes the target word as input and tries to predict the context words before and after it (Mikolov et al., 2013).

Word2Vec is formed by 3 main building blocks – (1) vocabulary builder, (2) context builder, and (3) neural network (Andrea, 2019). The vocabulary builder takes in the raw sentence data and extracts unique words to build a corpus (Bhanawat, 2019). Following that, the context builder converts the words to vectors by taking into consideration all the words in the range of the context window of the target word. Lastly, Word2Vec will use a neural network with 1 hidden layer for training, where the number of neurons corresponds to the embedding dimensions.

Implementation

Off-the-shelf pre-trained word embeddings are typically used for generic language use cases or when a dataset is too small to build meaningful custom embeddings. Given that the vocabulary size of our train dataset is approximately 20,400 with roughly 139,500 data points, it should provide a good learning scenario for a custom trained embedding layer (Anala, 2020).

Hyperparameters

To train the Word2Vec embeddings, we have set a minimum count of 2, embedding dimension of 300, and context window of 10. The minimum count means that words with one occurrence will be ignored by the model to prevent overfitting the trained embedding. Next, the embedding dimension was set to a high number as the quality for vector representations improves with an increase in the vector size (Pennington et al., 2014). A large embedding allows better differentiation of words as each dimension captures some aspect of the word meaning. However, research have shown that the quality of vectors starts to decrease after 300 dimensions which was why we have limited our embedding dimensions to 300. Lastly, the context window specifies how many words before and after the target word should be taken into consideration when training the embedding. As this value is typically set around 5 or 10, we have chosen the value 10.

4.2 GloVe

Global Vectors for Word Representation (GloVe)[3] is another unsupervised learning algorithm to obtain vector representation for words. In contrast to Word2Vec, it is a count-based model rather than a predictive model and incorporates both local and global context by using the word co-occurrence matrix (Ganegedara, 2019).

Algorithm

GloVe is a log-bilinear model based on the ratios of probabilities from the co-occurrence matrix and combines the intuition of a count-based model while capturing the linear structure utilised by methods such as Word2Vec (Pennington et al., 2014). It has a weighted least-squares objective that minimises the difference between the dot product of the vectors of two words and the logarithm of their probability of co-occurrences (Sciforce, 2018). The co-occurrence matrix is a $V \times V$ matrix where V is the vocabulary size (Khattak et al., 2019). Each entry is obtained by counting the number of times the particular vocabulary items occur together within the pre-specified context window while moving across the entire corpus. Since the matrix will be huge, we factorise it to yield a lower-dimensional matrix by minimising a "reconstruction loss", which tries to find lower-dimensional representations that can explain most of the variance in the high-dimensional data. From the matrix, the ratio of probabilities helps us distinguish relevant from irrelevant words and further differentiate between the two relevant words (Venugopal, 2021).

Implementation

To better compare different word embeddings, we have integrated pre-trained GloVe word embeddings into our model. More specifically, we have chosen to use the pre-trained 200-dimensions Twitter embeddings trained on 2 billion tweets with a vocabulary size of 1.2 million. These pre-trained word embeddings are a form of Transfer Learning as we are utilising the embeddings learnt and applying it to solve a similar task (Pai, 2020).

With our dataset also originating from social media, we believe that there could be some nuances captured within the embedding which may benefit model performance. With the larger dataset used to train the embedding, we believe that GloVe can potentially boost model performance due to an increased ability to capture the semantic and syntactic meaning of a word (Pai, 2020).

[3] GloVe word embedding accessible at https://nlp.stanford.edu/projects/glove/.

5. Model Building and Evaluation

In this section, we will be building different models and evaluating their performance in classifying suicidal text. In our problem statement, we are trying to predict a binary variable, specifically suicide or non-suicide.

The 5 models built range from machine learning, deep learning, to transformers. They include Logistic Regression (Logit), Convolutional Neural Network (CNN), Long Short-term Memory (LSTM), Bidirectional Encoder Representations from Transformers (BERT), and Efficiently Learning on Encoder that Classifies Token Replacements Accurately (ELECTRA) seen in Figure 12 below.

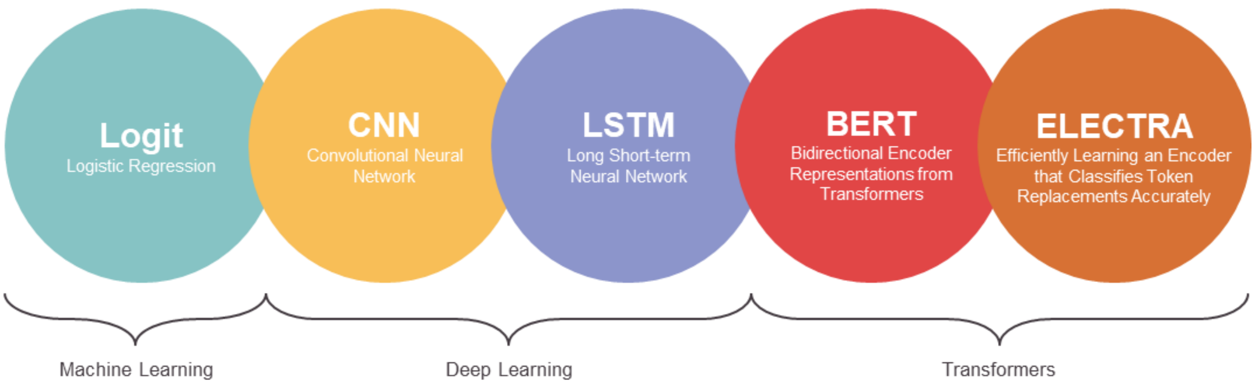


Figure 12. Models Built

The models were trained on the train dataset, finetuned on the validation dataset, and tested on the test dataset. We will be evaluating the model performance on the test dataset based on 4 evaluation metrics – Accuracy, Precision, Recall and F1 score. A greater focus shall be placed onto the F1 score for our use case to ensure a better application of our model in downstream tasks. Although false negatives are not ideal, F1 score provides a more balanced perspective of model performance as compared to recall.

5.1 Logistic Regression

In NLP, logistic regression is used as the baseline supervised algorithm for classification (Jurafsky and Martin, 2021). Moreover, it has a very close relationship with neural networks as it can be viewed as logistic regression classifiers stacked on top of each other (Jurafsky and Martin, 2021). We will be training Logistic Regression to serve as a baseline model for comparison purposes.

Model Architecture

Logistic regression consists of 4 components: (1) input feature representation, (2) classification function that computes the estimated class, (3) objective function to learn from the training dataset, and (4) algorithm for optimising the objective function.

For our logit model, as no finetuning is needed for the baseline model, we will not have a validation dataset and the dataset is split into train and test with a ratio of 8:2 instead. From the train dataset, the model will learn a vector of weights and bias term, where each weight will be associated with one of the input features, representing the importance of the feature to the classification decision. After learning the weights, we will test it on the test dataset and the classifier will multiply each input feature with its weight and sum up the weighted features with the bias term. Thereafter, the output will be passed through a sigmoid activation function to classify the text as suicidal or non-suicidal.

Model Variants

We have experimented with the following model variants, with variations made to the embedding layer:

- **Logit Model 1:** Custom Word2Vec Embeddings (300-dimensions)
- **Logit Model 2:** Pre-trained GloVe Embeddings (200-dimensions)

Hyperparameters

Default hyperparameters were used for the model.

Model Performance

The model performance for all Logistic Regression model variants can be seen in Table 1 below. The best model variant is Model 1 (Custom Word2Vec Embeddings) and it has outperformed Model 2 (Pre-trained GloVe Embeddings) on all metrics.

Logit Model Variants	Accuracy	Recall	Precision	F1 Score
1. Custom Word2Vec Embedding	**0.9111**	**0.8870**	**0.8832**	**0.8851**
2. Pre-trained GloVe Embedding	0.8774	0.8440	0.8394	0.8417

Table 1. Logit Models Performance Comparison

For regression and classification prediction problems, pre-trained word embeddings rarely perform as well as learning the word embeddings from the original dataset. This might also be due to suicide detection being a niche field and the difference in the training vocabulary and underlying corpus could have a strong influence on the resulting representation, making most off-the-shelf embeddings unsuitable. Additionally, off-the-shelf word vectors would only be the most effective in low-resource scenarios (Neubig et al., n.d.). However, our dataset is large enough to learn meaningful representations from customised embeddings and custom trained word embeddings would perform better especially in our niche use case.

5.2 Convolutional Neural Network (CNN)

As our project attempts to classify text data, the sequence of words plays a part in contributing to the connotation of a sentence. Previously, the Logistic Regression model was unable to capture this feature and hence we deployed several deep learning algorithms to process the sequences in our training data. According to Ce and Tie (2020), the Convolutional Neural Network (CNN) was proposed as an efficient way to classify text data as it is able to achieve decent prediction accuracy and consume lesser computational resources. As such, we have experimented with the CNN model as one of the deep learning algorithms in our project.

Model Architecture

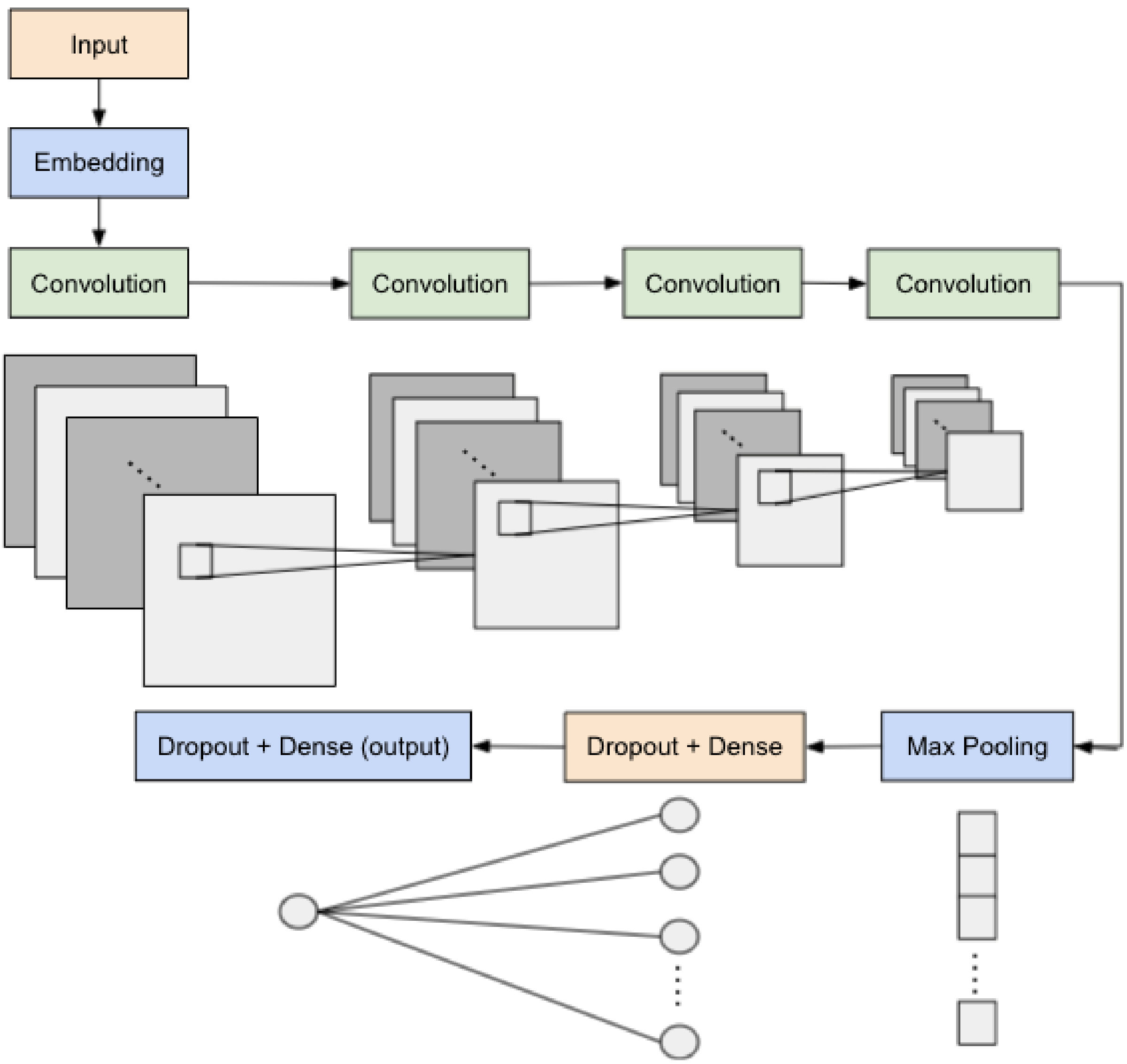


Figure 13. CNN Model Architecture

Our CNN model architecture consists of the following layers: an embedding layer, 4 convolutional layers, a pooling layer, a dropout layer, and a fully connected layer, as shown in Figure 13 above.

We first performed tokenization on our text data to convert each word into an integer. The embedding layer is responsible for mapping each word of the sentence in our training data into a feature vector of a fixed embedding size. The purpose of including an embedding layer is to represent word vectors in a semantically designed space, where words with similar meanings are close to each other, while words with different meanings are far apart from each other (Kim and Jeong, 2019). Next, we have the convolutional layers in place to extract contextual features from the input data. Earlier convolutional layers are expected to extract simple contextual information, while the later convolutional layers are responsible to capture key features and extract sentiments that will affect classification. After the convolutional layers, we applied the max-pooling technique as we want our model to focus on the most important features in the sentence, regardless of the ordering of words (Goldberg, 2015). Next, we added a dropout layer as a form of regularization in our CNN model to prevent overfitting. The final layer is a fully connected layer that connects all input and output neurons together. The vector passes through this layer and the output will go through a sigmoid activation function to classify whether the text is suicidal or non-suicidal.

<u>Model Variants</u>

We have experimented with the following model variants[4], with variations made to the Embedding layer:

- **CNN Model 1:** Random Initialisation (no pre-trained weights)
- **CNN Model 2:** Custom Word2Vec Embeddings (300-dimensions)
- **CNN Model 3:** Pre-trained GloVe Embeddings (200-dimensions)

<u>Hyperparameters</u>

We have experimented with different combinations to find the optimal hyperparameter values empirically and they are finalised as follows. We applied an embedding size of 300 for models 1 and 2, while model 3 has an embedding size of 200. The difference in embedding size was due to the dimension of loaded embedding weights into the embedding layer. Next, we stacked 4 convolutional layers and within each layer, there are 32 filters. We applied filter sizes of 5×5, 6×6, 7×7 and 8×8 in convolutional layers 1, 2, 3 and 4 respectively. We have also applied a dropout rate of 0.2. We adopted the Adam's optimiser with a learning rate of 0.00001 and set our loss function to be Binary Cross Entropy with Logits Loss (BCEWithLogitsLoss), which combines Sigmoid and Binary Cross Entropy Loss (BCELoss). The advantage of this loss function lies in the log-sum-exp trick, giving numerical stability in the values, which is not achievable by implementing the plain Sigmoid followed by BCELoss functions sequentially. Within each convolutional layer, we have also used the Rectified Linear Unit (ReLU) as the activation function to prevent the exponential growth of computation required in our neural network.

[4] CNN Model 1 has 9,588,857 trainable parameters, Model 2 has 249,857 trainable parameters, and Model 3 has 166,657 trainable parameters.

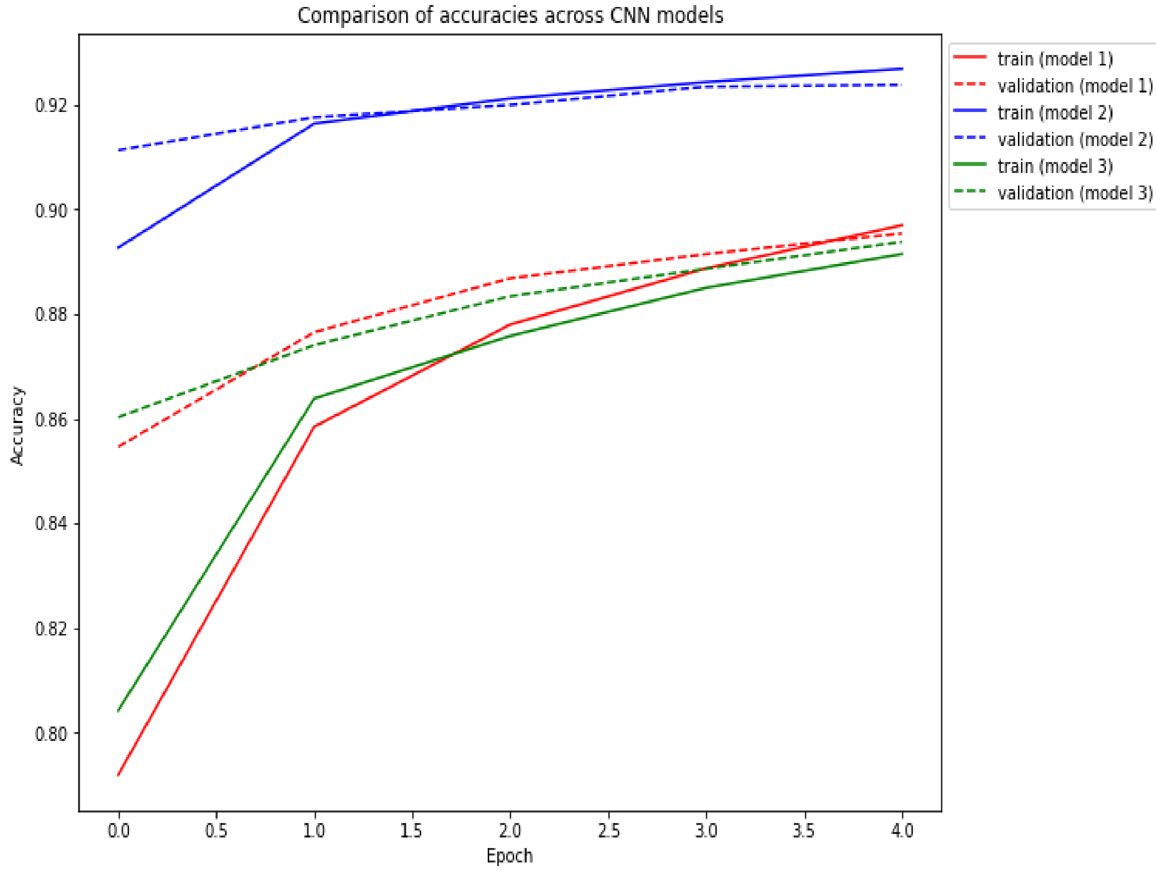


Figure 14. CNN Model Training Graph

Figure 14 above shows the training graph for all CNN model variations of train and validation accuracies across epochs. The accuracies have started to plateau as the number of epochs increases, indicating that the model training process was stable for all model variants.

Model Performance

The model performance for all CNN model variants can be seen in Table 2 below. The best model variant is Model 2 (Custom Word2Vec Embeddings) and it has performed the best across all metrics. This finding is aligned with the performance of the Logistic Regression model, where custom Word2Vec embeddings have performed the best.

CNN Model Variants	Accuracy	Recall	Precision	F1 Score
1. Random Initialisation	0.8985	0.8281	0.9010	0.8630
2. Custom Word2Vec Embedding	**0.9285**	**0.9013**	**0.9125**	**0.9069**
3. Pre-trained GloVe Embedding	0.9001	0.8511	0.8858	0.8681

Table 2. CNN Models Performance Comparison

5.3 Long Short-term Memory Network (LSTM)

Although CNN models are useful in text classification, they detect local and position-invariant key phrases and are unable to detect a long-range semantic dependency like Recurrent Neural Network (RNN) models (Minaee et al., 2021). RNNs works better than CNN with sequential data such as text, however, they suffer from the problem of vanishing gradients (Zhao et al., 2019). Thus, we have chosen to build a Long Short-term Memory Network (LSTM) model, a type of RNN model, which tackles the shortcomings of RNNs.

LSTM is capable of taking in longer sequences and remembering longer dependencies in a sequence. This is achieved due to its capability of retaining memory of relevant information and forgetting irrelevant information. More importantly, LSTM alleviates the vanishing gradient problem faced by RNNs through a memory cell in the LSTM layer to remember values over time intervals. This is achieved through a gating mechanism using the input, output and forget gates to control the flow of information into and out of the cell (Li et al., 2020).

Model Architecture

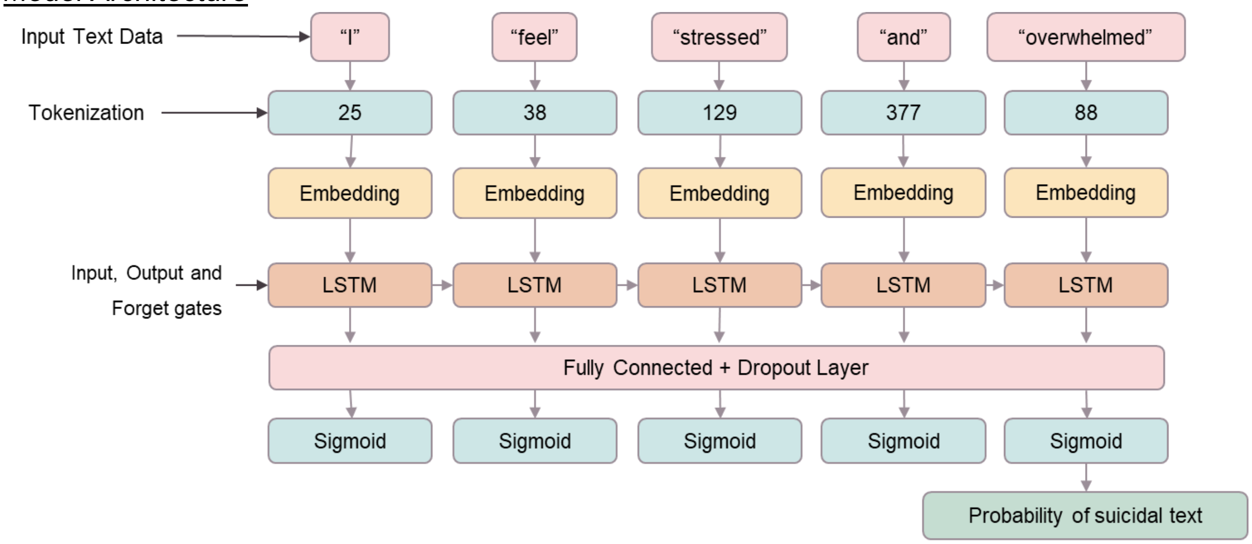


Figure 15. LSTM Model Architecture

Our LSTM network architecture consists of 5 layers as shown in Figure 15 above. Before the first layer, we performed tokenization to convert our words into integers. The first layer is the embedding layer, where the integer word tokens are converted into embedding vectors. The second layer is our LSTM layer where information is passed through and the gates within the LSTM model decides the information to forget, store or output. The third layer is a fully connected layer that is responsible to map the output of the LSTM layer to our output size of 1. The fourth layer contains the sigmoid activation function and it classifies whether the text is suicidal or non-suicidal. The final layer is the output layer where we took the output after the last time step and reshaped it such that the number of predictions equal to the batch size.

Model Variants
We have experimented with the following model variants[5], with variations made to the Embedding layer:

- **LSTM Model 1:** Random Initialisation (no pre-trained weights)
- **LSTM Model 2:** Custom Word2Vec Embeddings (300-dimensions)
- **LSTM Model 3:** Pre-trained GloVe Embeddings (200-dimensions)

Hyperparameters
Similar to CNN, we have experimented with different combinations to find the optimal hyperparameter values empirically and they are finalised as follows. The embedding size is 300 and we used a total of 128 hidden dimensions and 2 layers within the LSTM layer. The dropout rate has been set to 0.5. When compiling the model, we adopted the Adam's optimiser with a learning rate of 0.00001 and the BCEWithLogitsLoss loss function.

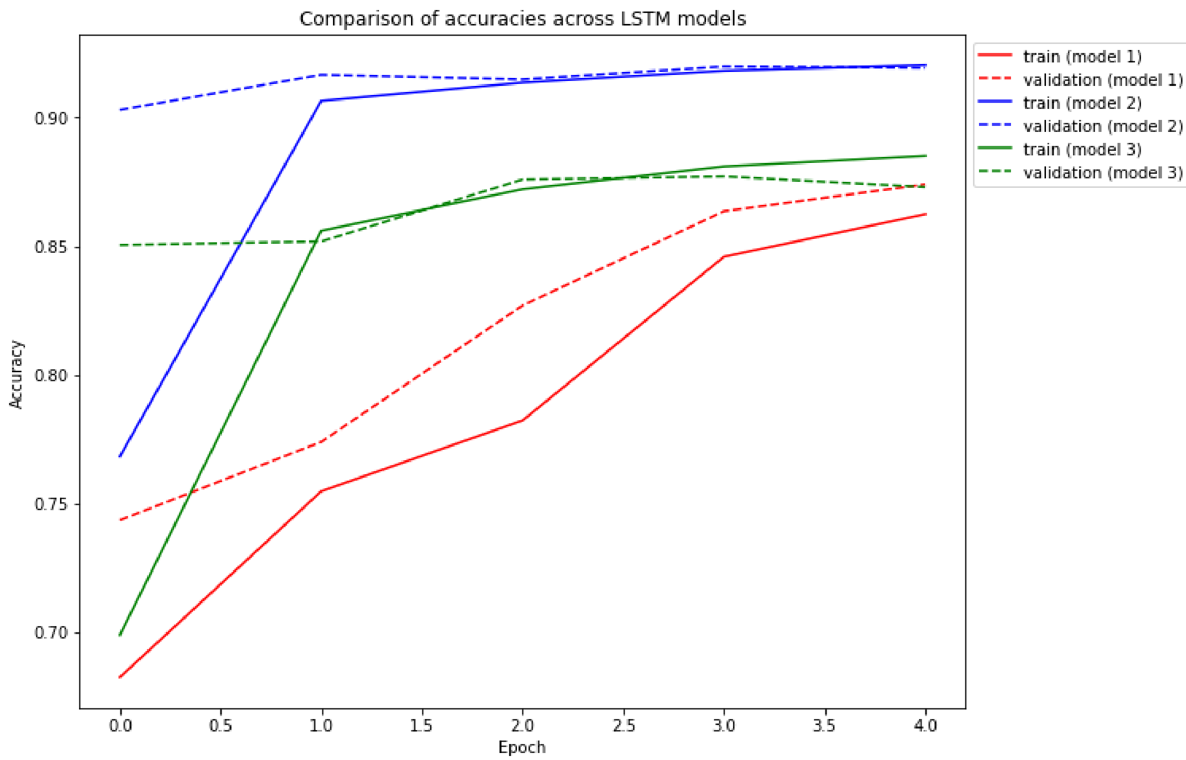


Figure 16. LSTM Model Training Graph

Figure 16 above shows the training graph for all LSTM model variations of train and validation accuracies across epochs. Similar to CNN, the accuracies have started to plateau as the number of epochs increases, indicating that the model training process was stable for all model variants.

[5] LSTM Model 1 has 9,691,385 trainable parameters, Model 2 has 352,385 trainable parameters, and Model 3 has 301,185 trainable parameters.

Model Performance

The model performance for all LSTM model variants can be seen in Table 3 below. The best model variant is Model 2 (Custom Word2Vec Embeddings) and it has performed the best across all metrics. This finding is aligned with the performance of the Logistic Regression and CNN models, where custom Word2Vec embeddings have performed the best.

LSTM Model Variants	Accuracy	Recall	Precision	F1 Score
1. Random Initialisation	0.8724	0.7982	0.8611	0.8285
2. Custom Word2Vec Embedding	**0.9260**	**0.8649**	**0.9386**	**0.9003**
3. Pre-trained GloVe Embedding	0.8825	0.7613	0.9206	0.8334

Table 3. LSTM Models Performance Comparison

5.4 Transformers

A transformer is a model with an attention-based encoder-decoder architecture (Cheng, 2020). The self-attention mechanism allowed better representation of words in NLP tasks, which have allowed various transformer model variations to achieve state-of-the-art performances in recent research advancements.

The transformer model considers the relationship between all words within a sentence, irrespective of their order. Transformers are able to handle long-range interactions better than deep learning models such as CNN and LSTM. Although LSTMs have bidirectional variants and are able to better capture long-term dependencies, vanishing gradient problems were elevated but not entirely eliminated (Culurciello, 2019). With a forget gate in LSTM, vanishing gradients will occur over long enough sequences.

Furthermore, traditional models are unable to be parallelized due to the sequential nature of inputs (Joshi, 2020). On the contrary, the absence of recursion allows transformers to parallelise tasks and accelerate processing time by processing sentences as whole using the attention mechanism and positional embeddings (Goled, 2021).

Most transformer models were pre-trained on large datasets using unsupervised techniques and subsequently fine-tuned for specific prediction tasks using supervised techniques. Transfer learning helps to vastly cut down on training time and significantly improves performance through more comprehensive representations as opposed to training from scratch. In this section, we aim to fine-tune two widely used transformers model, BERT and ELECTRA, to detect suicide text and compare their performance.

5.4.1 BERT

BERT, also known as *Bidirectional Encoder Representations from Transformers*, utilises the encoder structure of a transformer for language modelling and was developed by Google in 2018 (Devlin et al., 2018).

Model Architecture

BERT is pre-trained on 2 tasks, namely *Masked Language Model (MLM)* and *Next Sentence Prediction (NSP).* The MLM technique enabled bidirectional training in contrast to previous models which looked at a text sequence from a single-direction or combined bidirectional training.

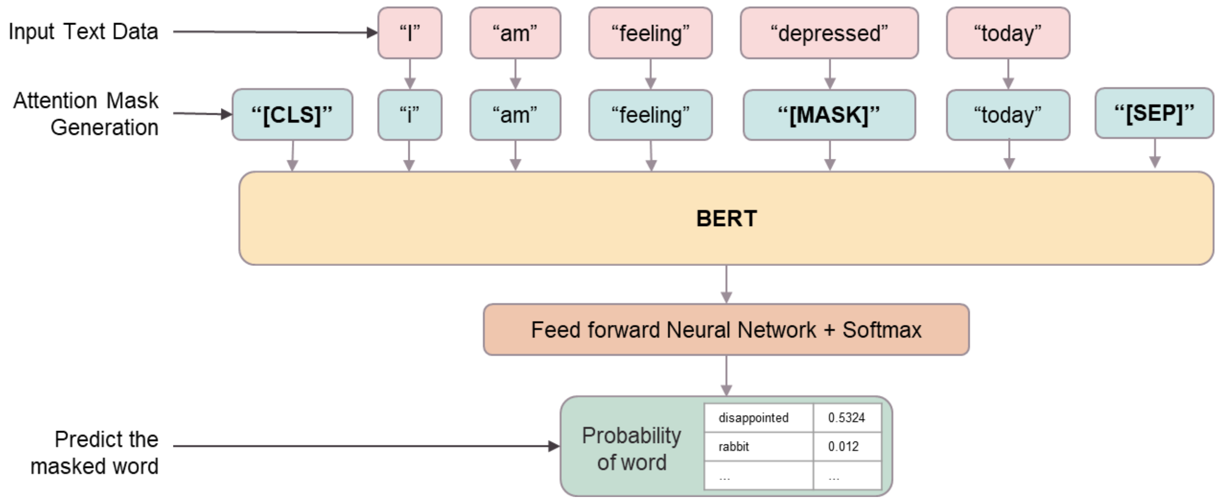


Figure 17. BERT Masked Language Model (MLM)

Referring to the example in Figure 17 above, the following input sentence "I am feeling depressed today" will be transformed into "I am feeling [MASK] today" after masking. The model is then trained to replace the masked tokens with the correct word, which has allowed the model to learn more accurate representations with the attention mechanism. MLM replaces 15% of the words in the sequences with a "[MASK]" token for training purposes.

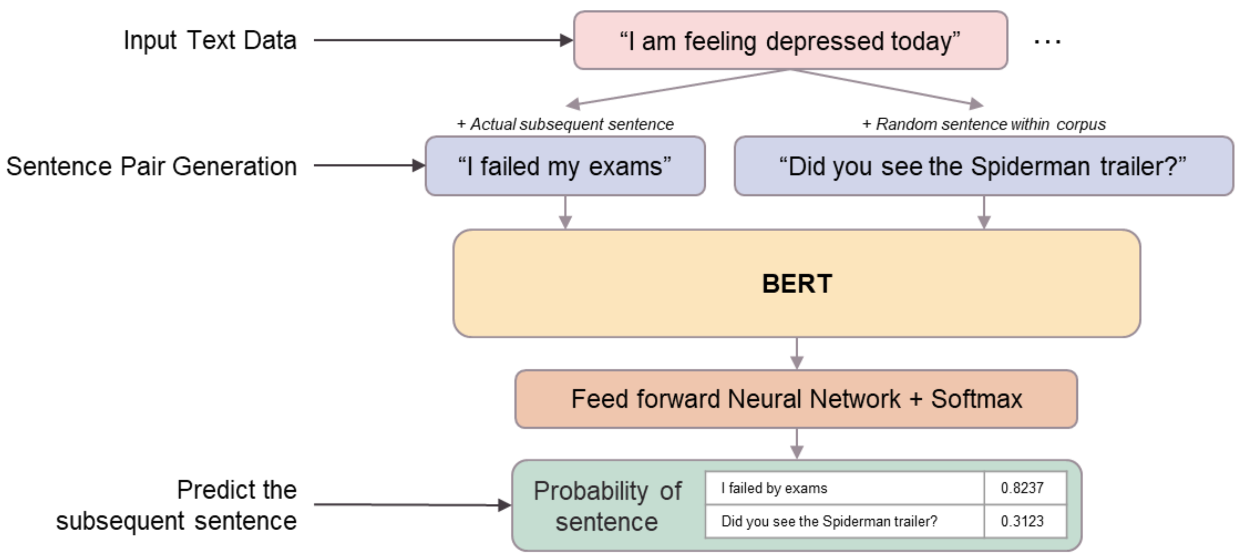


Figure 18. BERT Next Sentence Prediction (NSP)

MLM captures sentence features at a token level but does not capture sentence-level information, which is achieved by the NSP task incorporated. In the training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence. From Figure 18 above, we can see that "I failed my exams" is the actual subsequent sentence for the input text "I am feeling depressed today", while "Did you see the Spiderman trailer?" is a random sentence within the corpus. For each sentence, one real and one fake input will be used as the training data. Additional tokens “[CLS]” and “[SEP]” are used to denote the sentence boundary at the start and the end, respectively (Refer to Figure 17 above).

BERT was pre-trained on the English Wikipedia and BooksCorpus data, containing 3.3 billion words, and made available in 2 variations[6] — base and large.

Implementation
For our BERT model implementation, we utilised the BERT base model through the use of Hugging Face Transformers. Exhaustive experiments were conducted on a sample dataset to determine model hyperparameters and implementation techniques.

Better results were observed with the BERT large model but due to computational resource constraints, we have chosen the base model which still provided competent model performance. As BERT was originally trained on full sentences, the same should be applied to our input sentences and thus, a different data preprocessing technique was used. Using the same data points from previous models, we passed the original data into the Hugging Face BERT tokenizer. The tokenizer automatically maps the tokens to their corresponding token ids in the pretrained vocabulary into a training-friendly format, together with special tokens and attention masks generation.

To fine-tune a model, there are 3 main techniques commonly used: (1) Train the entire pre-trained architecture, (2) Train a few layers from the pre-trained architecture, or (3) Freeze the entire pre-trained architecture and train additional layers. The frozen pre-trained layers will not have their weights updated during fine-tuning and are commonly used to prevent overfitting. After testing empirically, we have decided to proceed with the first alternative of training the entire architecture as we have yielded better prediction results.

Model Variants
We have experimented with the following model variants, with variations in fine-tuning:

- **BERT Model 1:** Pre-trained
- **BERT Model 2:** Fine-tuned

Hyperparameters
The model fine-tuning was conducted on 1 epoch, with batch size of 6, and learning rate of 0.00001. According to Peltaron (n.d.), the general rule-of-thumb for fine-tuning should be around 1 or 2 epochs to prevent overfitting. Given the huge dataset size, 1 epoch should be sufficient with the extensive amount of data available to the model. Although a bigger batch size can accelerate model training, the limited memory constraint due to the long sequence text has restricted us with a small batch size. Lastly, the learning rate was deliberately set at a small number to prevent overfitting as we wish to readapt pretrained weights in an incremental manner.

[6] BERT base is a 12-layer, 768-hidden, 12-heads, 110M parameter neural network architecture. BERT large is a 24-layer, 1024-hidden, 16-heads, 340M parameter neural network architecture.

With a large learning rate, large gradient updates during fine-tuning can destroy the pre-trained features (Chollet, 2020).

Model Performance

The model performance for all BERT model variants can be seen in Table 4 below. The best model variant is Model 2 (Fine-tuned BERT) and it has outperformed Model 1 (Pre-trained BERT) on all metrics significantly.

BERT Model Variants	Accuracy	Recall	Precision	F1 Score
1. Pre-trained BERT	0.4681	0.9295	0.4156	0.5744
2. Fine-tuned BERT	**0.9757**	**0.9669**	**0.9701**	**0.9685**

Table 4. BERT Models Performance Comparison

Similar to the different feature representation techniques we have tried, pre-trained models are unlikely to perform as well as a custom trained model tailored for our specific use case. Interestingly, pre-trained BERT has an extremely high recall value but a subpar F1 score, which indicates that it predicts most inputs as positive. Fortunately, by placing our focus on F1 score, there is a more balanced evaluation of the model performance which renders pre-trained BERT unsatisfactory.

5.4.2 ELECTRA

ELECTRA, alternatively known as *Efficiently Learning an Encoder that Classifies Token Replacements Accurately*, is one of the latest pre-trained transformer models released by Google in 2020 (Clark et al., 2020). Most enhanced transformer models from BERT, such as RoBERTa and XLNet, leverages on larger datasets which require larger computational power and longer training time. ELECTRA on the other hand outperforms the aforementioned models on a few benchmark datasets with less than a quarter of the required computation power.

Model Architecture

ELECTRA proposed a new pre-training task called *Replaced Token Detection (RTD)* tackling the disadvantages of the MLM technique from BERT which corrupts the input with masked tokens. BERT only learns to predict a small subset of tokens (the masked tokens) instead of predicting every single input token, which reduces the amount learnt from each sentence. Generally, BERT still produces good results after transferring to downstream NLP tasks but require large amounts of computation (Clark et al., 2020)

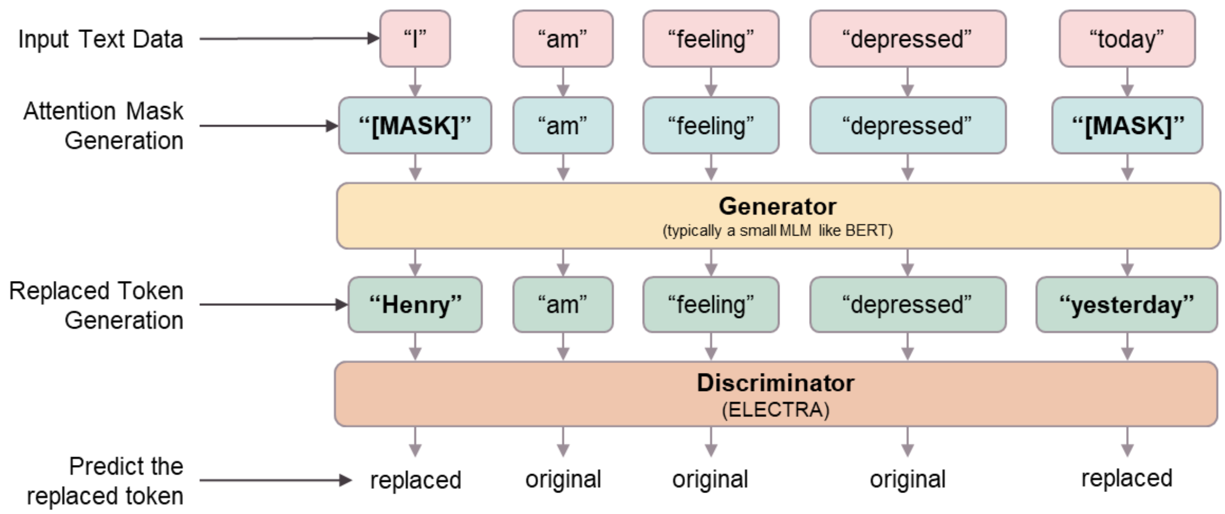


Figure 19. ELECTRA Replaced Token Detection (RTD)

On the contrary, ELECTRA uses RTD to train a bidirectional model where tokens are replaced with incorrect but plausible fakes instead of "[MASK]" tokens. Referring back to the same example given earlier in the BERT section, the input sentence "I am feeling depressed today" will be transformed into "Henry am feeling depressed yesterday", as seen in Figure 19 above. As compared to sentences replaced with "[MASK]" tokens, the replaced sentence makes more sense but still does not fit entirely within the context.

Subsequently, the discriminator attempts to identify tokens which have been replaced — inspired by generative adversarial networks (GANs) to distinguish between real and fake input data. This binary classification task encourages for a more accurate representation learning from all input positions as the model must learn an accurate data representation to solve the task. Additionally, ELECTRA requires less training data to achieve the same model performance due to its ability to receive more information per example.

The replacement tokens are generated from a generator, typically a small MLM like BERT. The generator is then trained jointly with the discriminator while sharing the same input word embeddings.

ELECTRA, the discriminator model, was made available in 3 variations[7] — small, base, and large. The small and base model were pre-trained on the same BERT data, while the large model was pre-trained on the XLNet data — an extension of the BERT dataset to 33 billion words with data from ClueWeb, CommonCrawl, and Gigaword.

[7] ELECTRA small is a 12-layer, 256-hidden, 4-heads, 14M parameter neural network architecture.
ELECTRA base is a 12-layer, 768-hidden, 12-heads, 110M parameter neural network architecture.
ELECTRA large is a 24-layer, 1024-hidden, 16-heads, 335M parameter neural network architecture

Implementation

For our ELECTRA model implementation, we are utilising the ELECTRA base model through Hugging Face Transformers, similar to BERT. We have chosen the base model as a similar sized network will allow a better comparison between the performance of the 2 transformer models. Similarly, the original data was passed into the Hugging Face ELECTRA tokenizer where the respective preprocessing takes place.

Model Variants

We have experimented with the following model variants, with variations in fine-tuning:

- **ELECTRA Model 1:** Pre-trained
- **ELECTRA Model 2:** Fine-tuned

Hyperparameters

The same training hyperparameters as the BERT model were utilised, where the model fine-tuning was conducted on 1 epoch, with batch size of 6, and learning rate of 0.00001.

Model Performance

The model performance for all ELECTRA model variants can be seen in Table 5 below. The best model variant is Model 2 (Fine-tuned ELECTRA) and it has outperformed Model 1 (Pre-trained ELECTRA) on accuracy, precision, and F1 score significantly. On the other hand, Model 1 (Pre-trained ELECTRA) has a higher recall score which was close to 1, but it does not affect our model choice since our focus is placed on F1 score.

ELECTRA Model Variants	Accuracy	Recall	Precision	F1 Score
1. Pre-trained ELECTRA	0.4025	**0.9908**	0.3918	0.5615
2. Fine-tuned ELECTRA	**0.9792**	0.9788	**0.9677**	**0.9732**

Table 5. ELECTRA Models Performance Comparison

Transformers Comparison

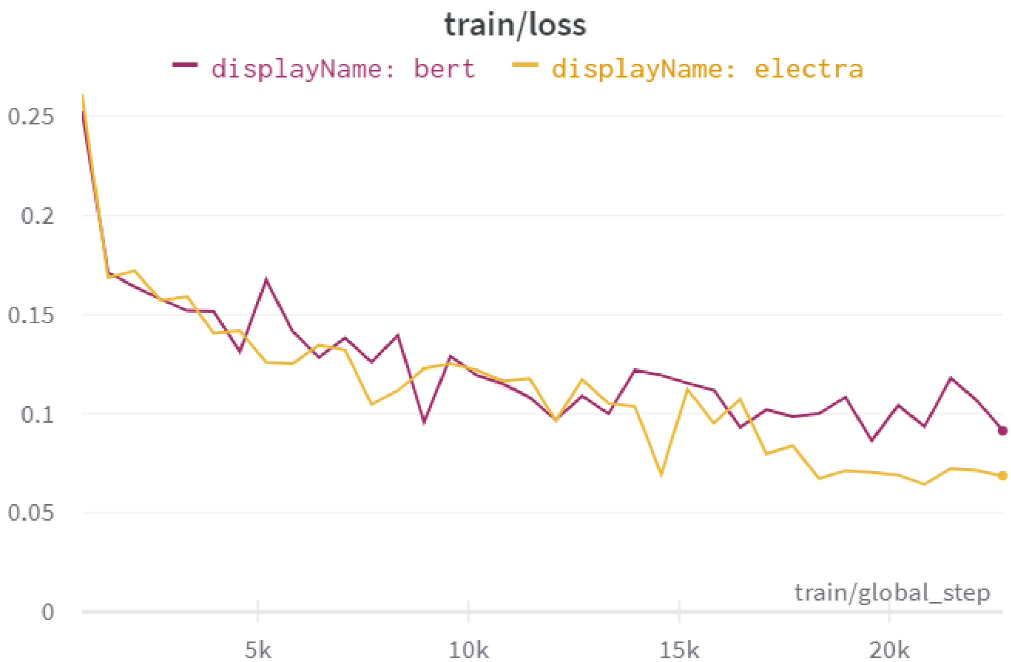


Figure 20. Transformers Model Training Graph

Figure 20 above shows the training graph for fine-tuned BERT and ELECTRA of training loss over 1 epoch, with 500 data points representing 1 step. In general, we could see similar decreasing trends with fluctuations between both models as they converge. At the end of the training, ELECTRA had a smaller loss in comparison to BERT which translates into better model performance in the evaluation metrics as observed in Tables 4 and 5 above. As the training loss has started to plateau and stabilise at the end of the epoch, we can conclude that the training on 1 epoch was adequate.

6. Model Selection

Table 6 below summarises the key results of the models executed. The best variations for Logistic Regression, CNN, and LSTM are using the custom trained Word2Vec embeddings, while BERT and ELECTRA are fine-tuned on our dataset.

Best Models	Accuracy	Recall	Precision	F1 Score
Logistic Regression	0.9111	0.8870	0.8832	0.8851
Convolutional Neural Network (CNN)	0.9285	0.9013	0.9125	0.9069
Long Short-Term Memory Network (LSTM)	0.9260	0.8649	0.9386	0.9003
BERT	0.9757	0.9669	**0.9701**	0.9685
ELECTRA	**0.9792**	**0.9788**	0.9677	**0.9732**

Table 6. Models Performance Comparison

The transformer models have outperformed the other models for all metrics. ELECTRA has achieved the highest F1 score, accuracy and recall score. This is followed up closely by BERT which has the highest precision score. Although the performance of these two models matches up with one another, ELECTRA has been proven to perform better and faster than BERT with the proposed RTD directly tackling the weakness of using MLM with BERT. Thus, our final model selection will be **ELECTRA**.

7. Chatbot Integration

With suicide rates rising among the youths in Singapore, more needs to be done to provide youths with help and support. As social stigma against mental health still remains, youths may be hesitant to reach out for help and prefer anonymous methods to avoid judgements (Goh et al., 2021). Additionally, youths tend to avoid seeking help as they are not comfortable confiding in a professional setting as they do not trust a stranger enough to share personal information and experiences related to their mental health condition (Radez et al., 2021).

According to Martínez-Miranda (2017), chatbots can help in the early detection of suicidal behaviours and provide support to individuals at risk. As an online conversational agent, chatbots can be a helpful tool for building conversations with youths behind a screen, eroding the fear that deter them from seeking help.

Model Architecture
Chatbots are considered *neural response generation* (NRG) models, which aim to predict a response based on input text. The two main types of chatbots consist of (1) *retrieval-based methods* which use heuristics to select a response from a collection of predefined responses and (2) *generative methods* which use machine learning to generate responses from scratch.

To achieve our objective, we will be using a combination of the aforementioned methods to build a mental health chatbot — generative methods to generate conversational responses and retrieval-based methods to provide suitable responses to suicidal text messages.

Implementation
Our chatbot was implemented using transformers as the self-attention mechanism can help to yield better performance as compared to traditional recurrent methods shown in previous sections.

The generative component of the chatbot uses DialoGPT, a large-scale pretrained dialogue response generation transformer-based model for multi-turn conversations developed by Microsoft in 2019. DialoGPT was inherited from the GPT-2[8] model and was trained on 147 million multi-turn dialogue from Reddit discussion threads. Similar to the previous transformer models, DialoGPT is made available in three variations[9] — small, medium, and large.

The original pre-trained DialoGPT model derived from the Hugging Face Transformers library was utilised. We have experimented with customised training of a chatbot but were unable to achieve comprehensible and relevant responses due to the lack of a huge chat dataset and training resources. On the other hand, responses generated from DialoGPT were comparable to human

[8] GPT-2, also known as Generative Pre-trained Transformer 2, is a large-transformer based language model released by OpenAI in 2019. GPT-2 was pre-trained on a dataset of 8 million web pages and the model aims to predict the next word, given all of the previous words within some text.
[9] DialoGPT small is a 12-layer, 117M parameter neural network architecture. DialoGPT medium is a 24-layer, 345M parameter neural network architecture. DialoGPT large is a 36-layer, 762M parameter neural network architecture.

response quality according to single-turn conversation Turing test conducted (Zhang et al., 2019). However, the pre-trained DialoGPT chatbot is unable to produce suitable responses to suicidal messages as it was not trained for such a purpose.

Thus, we have customised a retrieval-based component within the chatbot to suit our use case. We have compiled a library of comforting messages from various suicidal prevention websites together with a list of local helplines which users can reach out to for immediate support. For every user input, the message is checked for suicidal intent and if identified, the chatbot will respond with the appropriate message to provide comfort and help information.

```
==== Hello! I am Alex and I am your virtual friend. If you need a listening ear, I'm always here. To end the chat, input 'exit' in the chatbox. ====
>> You: hello
```

Alex: Hiya

`>> You: i need a friend to talk to`

Alex: I can be your friend

`>> You: i am feeling very sad recently`

Alex: I am sorry to hear that

`>> You: i don't feel like living anymore...`

Alex: Is there anything I can do to make this day go easier for you?

In times of severe distress where you need to speak with someone immediately, these are suicide hotline services available for you. You will be speaking with volunteers or professionals who are trained to deal with suicide crisis. Samaritans of Singapore (SOS; 24 hours): 1800 221 4444 Mental Health Helpline (24 hours): 6389 2222 Singapore Association for Mental Health (SAMH) Helpline: 1800 283 7019

`>> You: exit`

Alex: See ya

Figure 21. Mental Health Chatbot

Figure 21 above shows an example of our mental health chatbot which is currently accessible through the use of our Jupyter notebook script.

Limitations

With the growing capabilities of chatbots, we have seen an increase in adoption across various sectors such as customer and financial services in recent years. However, there are still limitations for chatbots to provide adequate empathetic responses in a natural human dialogue, which is the key for individuals seeking help (Robinson et al., 2018). Due to this limitation, chatbots might produce inappropriate responses at critical times or even mispredict suicidal risks. Furthermore, data privacy for personal health information is also a huge concern; measures must be taken to prevent such sensitive data from being compromised.

Although chatbots are unable to fully substitute health care and therapy, new approaches are required to tackle the rising suicide rates among young people. More research is required in this field before widespread adoption of mental health chatbots but we strongly believe they have the potential to be part of the solution.

8. Future Improvements

8.1 Business Improvements

8.1.1 Building a Multilingual Chatbot

According to a study assessing the usability of mental health care chatbots (Cameron et al., 2019), participants suggested that a chatbot useful feature would be to accept inputs of different languages, which is believed to promote more interaction with users.

Our chatbot currently only supports English conversations and the incorporation of more languages can expand our reach to a wider international community. This could be achieved by training and fine-tuning our models on conversational data in different languages, which can be compiled by scraping various social media platforms. Alternatively, we could adopt readily available models from the online community, such as the DiGPTame model (Cooper, n.d.), which is a DialoGPT model fine-tuned on Spanish data. However, this enhancement is only applicable to widely used languages as obtaining sufficient conversational data for dialects remains a challenge.

8.1.2 Integration of Chatbot onto Social Media Platforms

According to Statista Research Department (2021), social media usage is one of the most popular online activities and it is projected to see a total of 4.41 billion users in 2025. It was also found that social media is a source of social support for the isolated and lonely individuals during the COVID-19 pandemic. With the increase in social media usage, it was proposed by several researchers as a channel to detect mental health symptoms (Torous et al. 2021). Hence, we could possibly integrate our chatbot on a popular social media platform to increase its visibility and usage.

Facebook is an example of a popular social media platform as it currently has about 2.91 billion monthly active users (Statista Research Department, 2021). Besides having a large user base, Facebook has also attempted to flag content that may be related to suicide (Torous et al., 2021).

These flagged contents are currently communicated to the law enforcement team and there have been critiques on the privacy standards as Facebook as failed to seek consent from users to provide such details to a third-party person (Goggin, 2019). Though such privacy concerns still linger in our chatbot proposal, it is fundamentally different as compared to engaging a law enforcement team. This is due to the absence of a third-party human being managing mental well-being data. In addition, we posit such process, even with consent granted, will be less efficient than our chatbot. This is because the involvement of a third-party would usually require more waiting time before concrete help can be provided, whereas our chatbot can readily reach out to the account holder on Facebook. Hence, we proposed the integration of our chatbot on Facebook to provide timely assistance and reduce the controversy of privacy concerns.

8.2 Technical Improvements

8.2.1 Semi-supervised Learning to Improve Data Quality

When creating the dataset used in this project, the author made bold assumptions on the data labels – categorising all posts from the *SuicideWatch* subreddit as suicidal, while posts from the *teenagers* subreddit as non-suicidal. Therefore, the labels might not be an accurate representation.

To improve the quality of the data through more accurate labels, we can employ semi-supervised learning techniques to conduct pseudo labelling. With a small amount of human annotated data, it can be used to train deep neural networks in a supervised manner. Following which, the model can be used to generate labels for unlabelled data (Shubham, 2017). A combination of the human-labelled and pseudo-labelled data can be used to train our models for suicidal prediction.

By improving the quality of data fed into the model, we can ensure that the model built is more robust. Although pseudo labelled data may be slightly less accurate as compared to human annotated data, it is able to significantly cut down the number of manual hours required, which is its main advantage.

8.2.2 Larger Transformers Models to Improve Model Performance

Our current best-performing model is the ELECTRA base model fine-tuned on our dataset. We did not manage to integrate the ELECTRA large model due to computational constraints. As larger-sized transformer models were pre-trained with more data, layers, and parameters, they are proven to yield better results on benchmarks datasets such as GLUE[10] (Clark et al., 2020). Although larger transformer models can be utilised to improve the model's performance, it should also be noted that they require longer training time and may overfit on the training dataset.

8.2.3 Reinforcement Learning to Improve Chatbot Response

To consistently improve our chatbot capabilities and responses, we can potentially integrate reinforcement learning. Like how humans learn from interacting with our environment, reinforcement learning allows the chatbot to learn by interacting with its environment, the end users.

After each conversation, the chatbot makes use of data from their conversations with end users and observes the results from its action. Then, it accumulates rewards which can be modelled as ratings collected from end users after chatting with the bot. The bot's learning objective is to maximise the total rewards gained, which allows it to continuously improve on its ability to generate more appropriate responses. This way, the chatbot become increasingly efficient at responding to end users by learning from human feedback and developing its own control mechanism (Mnasri, 2019). However, the process of learning requires an abundance of user conversations, which can take a long time before it can achieve favourable performance.

[10] General Language Understanding Evaluation (GLUE) is a benchmark dataset commonly used to evaluate and analyse natural language understanding systems.

9. Conclusion

The prevalence of depression among youths in Singapore remains a serious social issue and early detection is crucial for timely intervention. Through our project, we have achieved our objective of detecting suicidal text on social media posts and have built models which achieved the **best F1 score of 0.9732** with **ELECTRA**. Subsequently, we have integrated the detection model into a functional chatbot, allowing us to reach out to individuals in need. Moving forward, we hope to enhance the performance for our detection model and chatbot, as well as expanding our reach to more people in need.

References

Anala, S. (2020, October 26). A Guide to Word Embedding. *Towards Data Science.* Retrieved from: https://towardsdatascience.com/a-guide-to-word-embeddings-8a23817ab60f

Andrea C. (2019, September 11). How to train the word2vec model. *Towards Data Science.* Retrieved from: https://towardsdatascience.com/how-to-train-the-word2vec-model-24704d842ec3

Bhanawat, V. (2019, June 28). The Architecture of Word2Vec. *Medium.* Retrieved from: https://medium.com/@vishwasbhanawat/the-architecture-of-word2vec-78659ceb6638

Biswas, D. (2020, September 15). Self-improving chatbots based on Deep Reinforcement Learning. *Medium.* Retrieved from: https://towardsdatascience.com/self-improving-chatbots-based-on-reinforcement-learning-75cca62debce

Brownlee, J. (2020, January 14). A gentle introduction to imbalanced classification. *Machine Learning Mastery.* Retrieved from: https://machinelearningmastery.com/what-is-imbalanced-classification/

Cameron, G., Cameron, D., Megaw, G., Bond, R., Mulvenna, M., O'Neill, S., Armour, C., McTear, M. (2019). Assessing the Usability of a Chatbot for Mental Health Care. Retrieved from: https://pure.ulster.ac.uk/ws/files/76756808/Internet_Science_chapter.pdf

Ce, P., Tie, B. (2020, November 10). An Analysis Method for Interpretability of CNN Text Classification Model. Retrieved from: https://www.mdpi.com/1999-5903/12/12/228/pdf

Cheng, R. (2020, July 22). *BERT Text Classification Using Pytorch*. Medium. Retrieved November 11, 2021, from https://towardsdatascience.com/bert-text-classification-using-pytorch-723dfb8b6b5b.

Chollet, F. (2020, May 12). Keras Documentation: Transfer Learning & Fine-tuning. Keras. Retrieved November 14, 2021, from https://keras.io/guides/transfer_learning/.

Choo, C. (2018, December 11). *More people in Singapore have experienced a mental disorder in their lifetime, study finds. Today.* Retrieved from: https://www.todayonline.com/singapore/more-people-singapore-have-experienced-mental-disorder-their-lifetime-study-finds

Clark, K., & Luong, T. (2020, March 10). More efficient NLP model pre-training with electra. Google AI Blog. Retrieved November 14, 2021, from https://ai.googleblog.com/2020/03/more-efficient-nlp-model-pre-training.html.

Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. International Conference on Learning Representations. Retrieved from: https://arxiv.org/pdf/2003.10555.pdf.

Cooper, N.. (n.d.). Finetuned DialoGPT model on Spanish Conversations. *Hugging Face.* Retrieved from: https://huggingface.co/ncoop57/DiGPTame-medium

Culurciello, E. (2019, January 10). The fall of RNN / LSTM. Medium. Retrieved November 14, 2021, from https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/abs/1810.04805v2.

Ganesh Kudva, K.; Abdin, E.; Vaingankar, J.A.; Chua, B.Y.; Shafie, S.; Verma, S.K.; Fung, D.S.S.; Heng Mok Kwee, D.; Chong, S.A.; Subramaniam, M. The Relationship between Suicidality and Socio-Demographic Variables, Physical Disorders, and Psychiatric Disorders: Results from the Singapore Mental Health Study 2016. Int. J. Environ. Res. Public Health 2021, 18, 4365. https://doi.org/10.3390/ijerph18084365.

Garbe, W. (2017, May 7). 1000x faster Spelling Correction. *Towards Data Science.* Retrieved from: https://towardsdatascience.com/symspellcompound-10ec8f467c9b

Goggin, B. (2019, January 7). Inside Facebook's suicide algorithm: Here's how the company uses artificial intelligence to predict your mental state from your posts. *Insider.* Retrieved from: https://www.businessinsider.com/facebook-is-using-ai-to-try-to-predict-if-youre-suicidal-2018-12

Goh, Y. H., Ang, Q., Ng, G. (2021, July 25). Helping youth improve mental health amid rising pressures. *The Straits Times.* Retrieved from: https://www.straitstimes.com/singapore/health/helping-youth-improve-mental-health-amid-rising-pressures

Goldberg, Y. (2015, October 6). A Primer on Neural Network Models for Natural Language Processing. Retrieved from: https://arxiv.org/pdf/1510.00726.pdf

Goled, S. (2021, March 16). Why transformers are increasingly becoming as important as RNN and CNN? Analytics India Magazine. Retrieved November 14, 2021, from https://analyticsindiamag.com/why-transformers-are-increasingly-becoming-as-important-as-rnn-and-cnn/.

Horev, R. (2018, November 17). Bert explained: State of the art language model for NLP. Medium. Retrieved November 14, 2021, from https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

Joshi, P. (2020, July 26). Transfer learning NLP: Fine tune bert for text classification. Analytics Vidhya. Retrieved November 14, 2021, from https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/.

Jurafsky, D., Martin, J. (2021, September 21). Logistic Regression. Speech and Language Processing. Retrieved from: https://web.stanford.edu/~jurafsky/slp3/5.pdf

Kaggle. (2021). Suicide and Depression Detection. *Kaggle.* Retrieved from: https://www.kaggle.com/nikhileswarkomati/suicide-watch

Khattak, F. K., Jeblee, S., Pou-Prom, C., Abdalla, M., Meaney, C., Rudzicz, F. (2019). A survey of word embeddings for clinical text. Retrieved from: https://doi.org/10.1016/j.yjbinx.2019.100057

Kim, H., Jeong, Y. S. (2019 April 29). Sentiment Classification Using Convolutional Neural Networks. Retrieved from: https://doi.org/10.3390/app9112347

Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P. S., and He, L. (2020, October 26). A Survey on Text Classification: From Shallow to Deep Learning. Retrieved from: https://arxiv.org/pdf/2008.00364.pdf

Martínez-Miranda J. (2017). Embodied Conversational Agents for the Detection and Prevention of Suicidal Behaviour: Current Applications and Open Challenges. Retrieved from: https://doi.org/10.1007/s10916-017-0784-6

Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. Retrieved from: https://arxiv.org/abs/1301.3781

Minarr, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., Gao, J. (2021). Deep Learning Based Text Classification: A Comprehensive Review. Retrieved from: https://arxiv.org/pdf/2004.03705.pdf

Mnasri, M. (2019). Recent advances in conversational NLP: Towards the standardization of Chatbot building. Retrieved from: https://arxiv.org/pdf/1903.09025.pdf

National Youth Council Singapore. (2019, November). National Youth Survey. Retrieved November 14, 2021, from https://www.nyc.gov.sg/en/initiatives/resources/national-youth-survey/.

Pai, A. (2020, March 16). An Essential Guide to Pretrained Word Embeddings for NLP Practitioners. *Analytics Vidhya.* Retrieved from: https://www.analyticsvidhya.com/blog/2020/03/pretrained-word-embeddings-nlp/

Peltarion. (n.d.). Bert - text classification / cheat sheet. BERT - Text classification / cheat sheet. Retrieved November 14, 2021, from https://peltarion.com/knowledge-center/documentation/cheat-sheets/bert---text-classification-/-cheat-sheet.

Pennington, J., Socher, R., Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. Empirical Methods in Natural Language Processing (EMNLP) (p./pp. 1532--1543), http://dx.doi.org/10.3115/v1/D14-1162.

Potts, C., Ennis, E., Bond, R.B. et al. (2021). Chatbots to Support Mental Wellbeing of People Living in Rural Areas: Can User Groups Contribute to Co-design?. Retrieved from: https://doi.org/10.1007/s41347-021-00222-6

Promises Editorial Team. (2020, June 5). Youth Depression in times of COVID19 Pandemic: How Can We Help?. *Promises.* Retrieved from: https://promises.com.sg/youth-depression-in-times-of-covid19-pandemic-hbl-home-based-learning-how-can-we-help/

Qi, Y, Sachan, D., Felix, M., Padmanabhan, S., Neubig, G. (n.d.) When and Why are Pre-trained Word Embeddings Useful for Neural Machine Translation? *Language Technologies Institute.* Retrieved from: https://aclanthology.org/N18-2084.pdf

Radez, J., Reardon, T., Creswell, C., Lawrence, P. J., Georgina E., Waite, P. (2021). Why do children and adolescents (not) seek and access professional help for their mental health problems? A systematic review of quantitative and qualitative studies. Retrieved from: https://doi.org/10.1007/s00787-019-01469-4

Robinson, J., & Thorn , P. (2021, October 24). Do Chatbots have a role to play in suicide prevention? The Conversation. Retrieved November 14, 2021, from https://theconversation.com/do-chatbots-have-a-role-to-play-in-suicide-prevention-105291.

Sciforce. (2018, August 14). Word Vectors in Natural Language Processing: Global Vectors (GloVe). *Medium.* Retrieved from: https://medium.com/sciforce/word-vectors-in-natural-language-processing-global-vectors-glove-51339db89639

Shubham, J. (2017, September 21). Pseudo labeling: Semi supervised learning. Analytics Vidhya. Retrieved from: https://www.analyticsvidhya.com/blog/2017/09/pseudo-labelling-semi-supervised-learning-technique/

Statista Research Department. (2021, November 1). Number of monthly active Facebook users worldwide as of 3rd quarter 2021. *Statista.* Retrieved from: https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/

Statista Research Department. (2021, September 10). Number of social network users worldwide from 2017 to 2025 (in billions). *Statista.* Retrieved from: https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/

Tham, D. (2021, July 8). Suicide cases in Singapore highest in 8 years amid covid-19 pandemic. Channel News Asia. Retrieved November 14, 2021, from https://www.channelnewsasia.com/singapore/suicide-highest-record-elderly-mental-health-isolation-covid-19-1984716.

Thushan Ganegedara. (2019, May 5). Intuitive Guide to Understanding GloVe Embeddings. *Towards Data Science.* Retrieved from: https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010

Torous, J., Bucci, S., Bell, I. H., Kessing L. V., Jepsen, M. F., Whelan, P., Carvalho, A. F., Keshavan, M., Linardon, J., Firth, J. (2021, September 9). The growing field of digital psychiatry: current evidence and the future of apps, social media, chatbots, and virtual reality. Retrieved from: https://doi.org/10.1002/wps.20883

Vatsal. (2021, July 29). Word2Vec Explained. *Towards Data Science.* Retrieved from: https://towardsdatascience.com/word2vec-explained-49c52b4ccb71

Venugopal, K. (2021, May 8). Mathematical Introduction to GloVe Word Embedding. Retrieved from: https://becominghuman.ai/mathematical-introduction-to-glove-word-embedding-60f24154e54c

Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J. & Dolan, B. (2019). DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation. https://arxiv.org/abs/1911.00536.

Zhao, Y., Shen, Y., Yao, J. (2019). Recurrent Neural Network for Text Classification with Hierarchical Multiscale Dense Connections. Retrieved from: https://www.ijcai.org/proceedings/2019/0757.pdf