Detecting Lies in Text

Degree: MSc Advanced Computer Science

Author: Qin Liu, 2007582

Supervisor: Professor Yukun Lai

October 2022

School of Computer Science and Informatics

Cardiff University

Abstract

According to the research done by University of Massachusetts psychologist Robert S. Feldman, 60% of participants lied at least once during a 10-minute conversation and told two to three lies on average (UMass Amherst, 2002). Furthermore, with the rapid development of social media and chat tools nowadays, false information and lies are flooding the entire network. Thereby, deception detection in text has been growing rapidly in the past few years. The deception detection exists in numerous fields, ranging from tweets to stories and news, there are considerable different datasets and many different advanced models for deception detection.

The purpose of this study is to gather and analyse data from different fields of life, in an attempt to discover patterns of texts that contain false information or deception and try to develop some machine learning models incorporating neural networks that would be able to identify such texts. Finally, to figure out whether the features extracted from the model trained on one dataset can generalise and work on other datasets.

The study found that BERT performs the best out those text classification methods, followed by random forest based on sentiment feature, bag-of words with TF-IDF feature, word count feature, and part-of-speech feature together. Also, linear regression has a good performance. Moreover, the trained BERT can provide some useful features to generalise to another dataset and work really well.

Acknowledgements

I would like to thank Prof. Yukun Lai for his guidance and assistance throughout this project and providing helpful recommendations and feedback at all phases of the development.

I would also like to thank my family and friends, for their help and support.

Table of Contents

Table of Figures and Tables

1. Introduction

Recent theoretical developments have revealed that in daily life, lying is well known for its ubiquity, with one study finding that 40% of people tell a lie at least once a day (Serota, Levine, Boster. 2010). Moreover, studies have shown that lies are widely shared across social media and online platforms. False news reaches more people and faster than the truth (Vosoughi et al. 2018).

However, people's ability to identify lies is not as good as the prevalence of lies, and Bond and DePaulo (2006) summarized 206 studies and found that people were only 47% correct in identifying lies, which is less than 50% of the random probability, and many other studies have obtained results that are consistent with this research (Bond & DePaulo. 2008). This creates a need for algorithms that will assist people in distinguishing between truth and deceptive messages.

In accordance with the review of existing studies, there has been less evidence for lie detection based on text. Thereby, the aims of the project are twofold: firstly, deceptive text can be detected using AI algorithms like Naïve Bayes, linear regression, support vector machine, and decision tree, and neural networks such as LSTM and Bert; secondly, AI algorithms can generalise across different datasets, allowing for models that can be re-used across different sources.

The main contribution this study offers is to choose the most appropriate model of basic classification methods and neural networks training on text dataset of true and false information, so that we can tell lie and truth from text automatically. Particularly, a BERT model has been trained using transfer learning, which generally means using what has been trained and stored to generalise on other datasets. Thus, we don’t need to collect tons of data for the better result, and the efficiency and performance would be significantly enhanced.

Furthermore, those optimisation strategies including adversarial training (Jin Yong, Yanjun.

2021) and EMA (Exponential Moving Average) (Müller, Rafael, et al. 2021) have been applied into LSTM and BERT.

2. Background

This section introduces the theoretical knowledge of applied techniques in the study and similar work, comprising evaluative measures, machine learning features, traditional machine learning models, deep learning techniques, and related work.

2.1 Evaluative Measures

The following metrics were used to evaluate the performance of our classifiers:

1. accuracy, defined as: $\frac{TP+TN}{TP+TN+FP+FN}$
2. precision, defined as: $\frac{TP}{TP+FP}$
3. recall, defined as: $\frac{TP}{TP+FN}$
4. F1 score, defined as: $2\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$

where TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative.

The metrics described above might provide varied insights depending on the categorization task at hand. For example, if we want to classify accurately, one of the metrics, accuracy, would be more significant than the others. Furthermore, if we want to decrease type II error (FN), recall could be a better statistic to use. Using a mix of these measures enables us to adapt our model based on our primary goal.

2.2 Machine Learning Features

2.2.1 TF-IDF

In text classification, the TF-IDF algorithm is often used for feature extraction. The TF-IDF algorithm is based on the frequent occurrence of single words in the corpus.

The second statistical method for judging the degree of importance, the main idea is to first count the term frequency (TF), and consider the occurrence of words

The more the number of times, the more positive relevance the document may have with the word, and the inverse document frequency (IDF) is used to reduce the weight of common words. The calculation formula is

$$\mathrm{TF}-\mathrm{IDF}_{i,j}=\mathrm{TF}_{i,j}\times\mathrm{IDF}_{i,j}$$

where TFIDF represents the product of term frequency TF and inverse document frequency IDF. The larger the value of TFIDF, the greater the importance of the current text.

The data used in this article are discretized keyword entries, and the clustering algorithm cannot directly calculate keywords. Therefore, this article uses the TF-IDF method to convert keywords into word frequency vectors. The clustering algorithm calculates the number of samples by calculating the distance between word frequencies. Similarity between

.

2.2.2 Bag-of-words Feature

Words or consecutive multiple words in the review text represent the bag-of-words feature of the text. Bag-of-words feature is also called n-gram feature, unigram, bigram, and trigram are commonly used bag-of-words features (Jindal N. et al. 2008). Statistics on the word frequency of words or phrases are also used to express bag-of-words features (Ott M. et al. 2013). The bag-of-words feature is a very effective feature in research fields such as opinion mining and sentiment analysis. In the direction of false comment detection, its effectiveness is higher than other text features, but there are obvious differences in detection effects in different datasets. For instance, on the dataset constructed by the crowdsourcing platform, the bag-of-words feature can reach an accuracy rate of nearly 90%, whilst on the comment dataset of the review website, only less than 70% accuracy rate is achieved (Arjun Mukherjee. 2013). Fake comments on review websites deliberately simulate real comments in terms of language and vocabulary. The use of the bag-of-words feature alone does not have a strong ability to recognize fake

comments. Combining with user behaviour characteristics can achieve a higher recognition accuracy.

2.2.3 Part-of-speech Feature

Part-of-speech features are extracted through part-of-speech tagging and frequency statistics of the text. Li et al. (2014) found that real reviews and fake reviews constructed by crowdsourcing have the following characteristics in terms of part of speech. The former contains more nouns, adjectives, prepositions, qualifiers, and conjunctions; while the latter contains more verbs, adverbs, pronouns, and antecedents. This is basically consistent with the early analysis of truthful writing and imaginative writing (Depaulo et al. 1996). However, false reviews fabricated by domain experts do not satisfy this rule. They contain more nouns, adjectives, and definite words than real reviews, while verbs and adverbs are less than real reviews. This is because domain experts have a stronger purpose of imitating real reviews when fabricating reviews, and the characteristics of imitating real reviews from details such as product information description and consumption experience are more confusing. There are certain differences in the distribution of part of speech in review texts in different fields, which is consistent with the research conclusion of computational linguistics, that is, the distribution of part of speech in a text is related to the text type (Rayson et al. 2001). However, when applied to the domain transfer problem of false reviews, its robustness is still better than the bag-of-words feature.

2.2.4 Sentiment Lexicon Feature

A sentiment lexicon (Kaity M. 2020) is one of the most valuable sentiment analysis resources for any language (Ahire S. 2015). They are important resources for both lexicon-based and machine-based learning approaches (Sun S. 2017), with many researchers using sentiment lexicons to create unsupervised sentiment models or as training features to train machine learning algorithms in supervised approaches (Giachanou A et al. 2016). A sentiment lexicon is a collection of words (also known as polar or opinion words) that have a positive or negative sentiment orientation (Medhat W. 2014). Wonderful, beautiful, and magnificent are examples

of positive sentiment adjectives. Negative feeling words, on the other hand, include horrible, poor, and bad.

2.3 Traditional Machine Learning Models

2.3.1 Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. According to Zhang (2004), Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Using the naive conditional independence assumption that

$$P(x_i \mid y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i \mid y)$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\downarrow$$

$$y = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

and we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$; the former is then the relative frequency of class y in the training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. (For theoretical reasons why naive Bayes works well, and on which types of data it does, see the references below.)

There are some popular types of Naïve Bayes classifier, such as multinomial Naïve Bayes, which is mostly used for document classification problem, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document (Rennie et al, 2003); Bernoulli Naïve Bayes, is similar to the multinomial naive bayes but the predictors are Boolean variables (McCallum et al, 1998). The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not (Manning et al, 2008); Gaussian Naïve Bayes, which refers to that when the predictors take up a continuous value and are not discrete (Metsis et al, 2006), we assume that these values are sampled from a gaussian distribution.

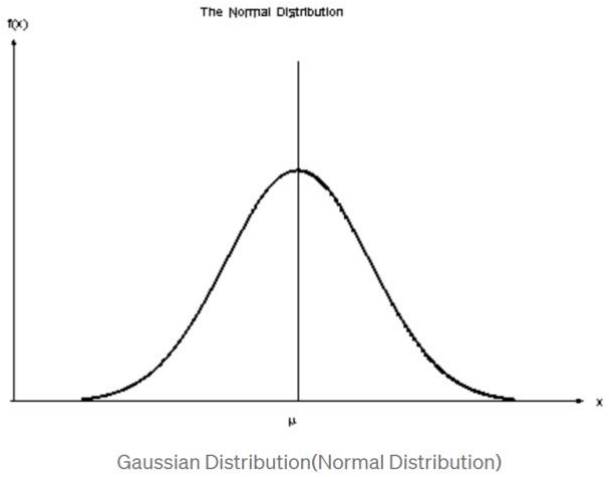


Figure 1: Gaussian distribution

Since the way the values are present in the dataset changes, the formula for conditional probability changes to

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

2.3.2 Linear Regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering the number of independent variables being used (Rencher et al 2012).

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between X (input) and Y (output). Hence, the name is Linear Regression. In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model. To consider the two classical linear regression model, simple linear regression and multiple linear regression.

Simple linear regression is an approach for predicting a response using a single feature. It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value (y) as accurately as possible as a function of the feature or independent variable (x) (Hilary,1967). According to David (2009), Multiple linear regression attempts to model the relationship between two or more features and a response by fitting a linear equation to the observed data. Clearly, it is nothing but an extension of simple linear regression. Consider a dataset with P features (or independent variables) and one response (or dependent variable). Also, the dataset contains n rows/observations.

2.3.3 Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features (Breiman, 1984). A tree can be seen as a piecewise constant approximation (Quinlan,1993). For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

Some advantages of decision trees are: Simple to understand and to interpret. According to the research of Hastie etc (2009), trees can be visualized. Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values. The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree. Able to handle both numerical and categorical data. However, scikit-learn implementation does not support categorical variables for now. Other techniques are usually specialised in analysing datasets that have only one type of variable. See algorithms for more information. Able to handle multi-output problems. Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret. Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model. Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

2.3.4 Random Forest

A random forest is a machine learning technique that's used to solve regression and classification problems. According to the researchers of the team of Hastie (2008) , it utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. A random forest algorithm consists of many decision trees. The 'forest'

generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms. The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

To the features of a Random Forest Algorithm, it can eradicate the limitations of a decision tree algorithm. It reduces the overfitting of datasets and increases precision. It generates predictions without requiring many configurations in packages (like scikit-learn). Therefore, it's more accurate than the decision tree algorithm, and can provide an effective way of handling missing data. Besides, it can produce a reasonable prediction without hyper-parameter tuning (Prinzie et al. 2008).

To consider the applications of random forest, it can refer to banking, health care, stock market and e-commerce. Random forest is used in banking to predict the creditworthiness of a loan applicant. This helps the lending institution make a good decision on whether to give the customer the loan or not. Banks also use the random forest algorithm to detect fraudsters (Piryonesi et al. 2019). Health professionals use random forest systems to diagnose patients.

Patients are diagnosed by assessing their previous medical history. Past medical records are reviewed to establish the right dosage for the patients. Financial analysts use it to identify potential markets for stocks. It also enables them to identify the behaviour of stocks. Through rain forest algorithms, e-commerce vendors can predict the preference of customers based on past consumption behaviour.

2.3.5 KNN

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique, which assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories

(Altman et al. 1992). Besides, this algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. It is also a non-parametric algorithm, which means it does not make any assumption on underlying data (Coomans.1982). And it can be called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

To think about both its advantages and its disadvantages: it can be simple and easy to implement, and there's no need to build a model, tune several parameters, or make additional assumptions. On the other hand, the algorithm is versatile, which means that it can be used for classification, regression, and search (as we will see in the next section). However, the algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase (Li et al. 2004).

2.4 Deep Learning Techniques

2.4.1 LSTM

Long short-term memory (Long short-term memory, LSTM) is a special kind of RNN, mainly to solve the problem of gradient disappearance and gradient explosion during long sequence training (S. Hochreiter, 1996).

Unlike RNN which only has a single tanh loop structure, LSTM is a special network structure with three "gates" structure, which will retain the state at each moment, which solves the traditional RNN loop unit's state in each cycle. The weakness of being covered and unable to deal with long-term dependence (Zhu X. et al. 2015, Pascanu R et al. 2013). The model structure of LSTM is shown in Figure 2.

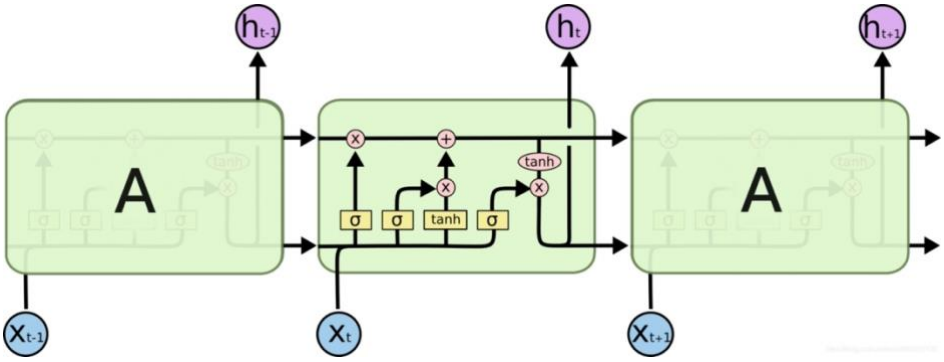


Figure 2: LSTM structure

The symbols in the figure above are defined as follows:

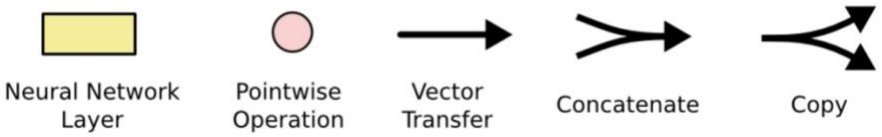
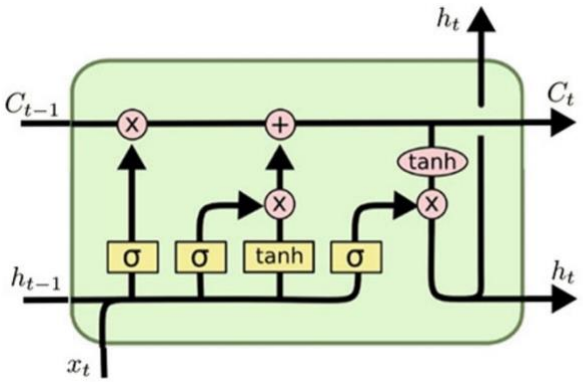


Figure 3: LSTM symbol definition

LSTM information flow:



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

Figure 4: LSTM information flow

An LSTM cell has 3 gates, which are called forget gate (f gate), input gate (i gate) and output gate (o gate). It should be noted that the output o_t of the output gate is not the final output of the LSTM cell, and the final output of the LSTM cell is h_t and C_t. σ represents the sigmoid activation function.

(1) Forget Gate

The forget gate determines what information is thrown away from the previous cell state (that is, how much information is retained), and selectively forgets the information in the previous cell state.

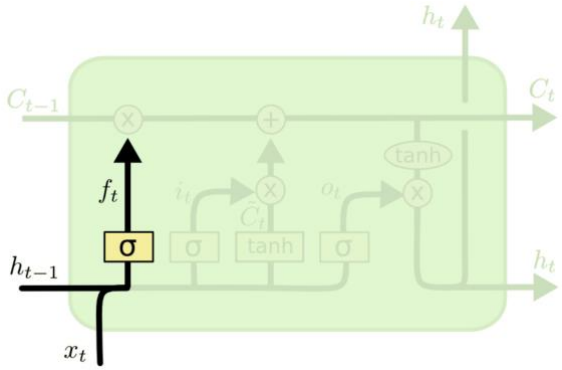


Figure 5: Forget gate

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

(2) Input Gate

The function of the input gate is to determine how much new information is added to the cell state. It is also realized through the sigmoid function. The input gate and the forget gate act simultaneously to form a new state.

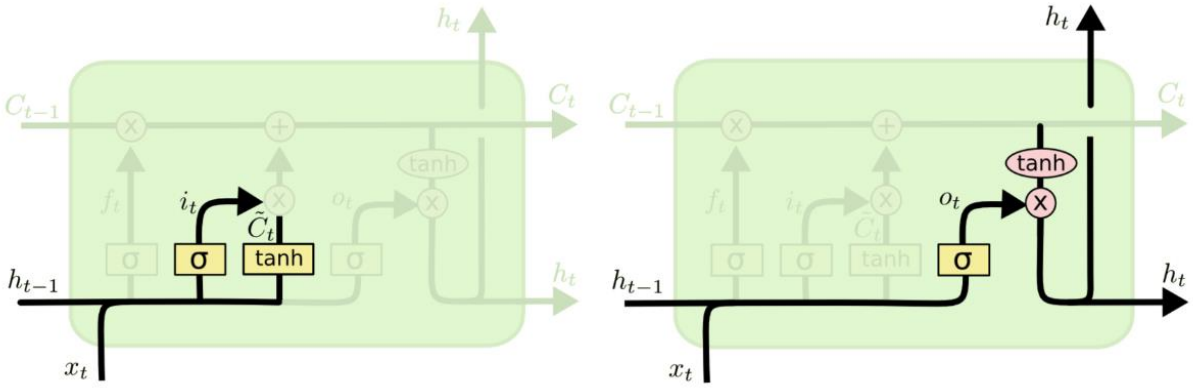


Figure 6: Input gate

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

(3) Output Gate

The function of the output gate is to decide what information to output. The output gate is also implemented by the sigmoid function.

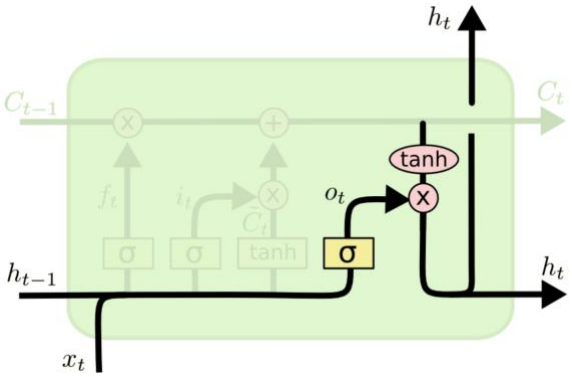


Figure 7: Output gate

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

In brief, LSTM can preserve long-term memory more effectively through the above-mentioned gate structure. Through the increased gate control structure, it fundamentally changes the forward propagation mode of traditional RNN and solves the problem of long-term dependence and vanishing gradient problem that traditional RNN cannot solve.

2.4.2 BERT

The BERT model (Jacob et al. 2019) can be understood as a general natural language understanding (Natural Language Understanding, NLU) model, which can be flexibly applied to a variety of different natural language processing tasks and has a good performance. For different application scenarios, you only need to add an additional input layer and fine-tune it according to the specific task, instead of modifying the model structure for a specific task. This is also one of the main advantages of the BERT model. The structure of the BERT model is shown in Figure 8, where E_n represents the nth token/symbol mark of the input sequence, T_{rm} is the transmission module, and Tn is the corresponding output embedding.

The bidirectional transformer feature (Vaswani et al. 2017) of the BERT model enables each word in the input sequence to be processed by the self-attention mechanism to obtain a new representation based on the weighted sum of all the word representations in the input sequence, and through the construction

The built network structure can learn representations that contain more contextual interaction information. Therefore, in this BERT based on a two-way transformer
Transfer learning on the model can learn high-quality embedded expressions containing false information.

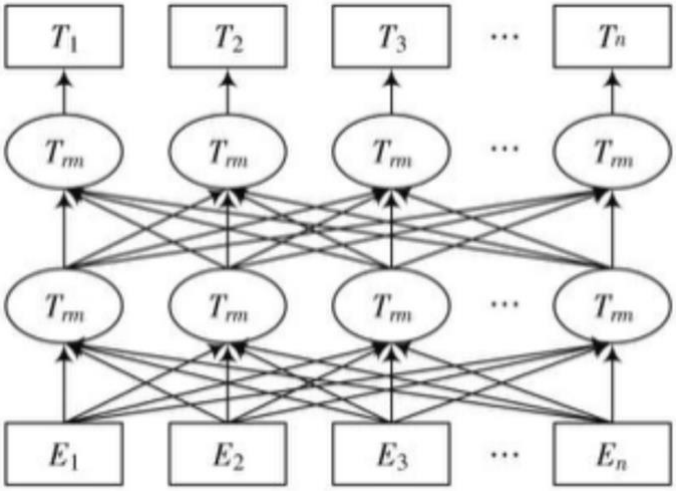


Figure 8: BERT structure

2.4.3 Transfer Learning

In 2005, the Broad Agency Announcement (BAA) 05-29 of Defense Advanced Research Projects Agency (DARPA)'s Information Processing Technology Office (IPTO) 2 gave a new mission of transfer learning: the ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks. In this definition, transfer learning aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task. In contrast to multi-task learning, rather than learning all of the source and target tasks simultaneously, transfer learning cares most about the target task. The roles of the source and target tasks are no longer symmetric in transfer learning (S. J. Pan et al. 2009).

Here is the classical process of transfer learning.

Definition 1: Domain: The domain is composed of sample feature space χ and feature distribution P(X), then the neighborhood $D=\{\chi,P(X)\}$, where $X=$ $\{x_1, x_2, \cdots, x_n\} \in \chi$. In transfer learning, domains are mainly divided into a source domain Ds and a target domain Dt. The source domain represents a domain that has been learned and acquired knowledge or skills, and the target domain represents a new domain that has not yet been learned.

Definition 2: Task: The task consists of the objective prediction function $F(\cdot)$ and the label space μ constitutes, that is, $T = \{\mu, F(\cdot)\}$. In transfer learning, the objective function $F(\cdot)$ is obtained from the data training of the source domain *Ds*, and the dataset is $\{xi, yi\}$, where $xi \in X$ and $yi \in \mu$. The objective function $F(\cdot)$ can be used to predict the label of the target domain *Dt*.

Definition 3: Transfer learning: Given a labelled source domain $Ds = \{xi, P(xi)\}^n i= 1$ and task Ts, and an unlabelled target domain $Dt =$

$\{xj, P(xj)\}^n j=^+ n^m + 1$

and task Tt, where $Ds \neq Dt, P(Xs) \neq P(Xt)$,

$Ts \neq Tt$. Transfer learning actually refers to the objective function $F(\cdot)$ and the label space μ obtained by learning and training the source domain Ds or task *Ts* to predict the label of the target domain *Dt* or the learning and training task *Tt*. The process of transfer learning is shown in Figure 9.

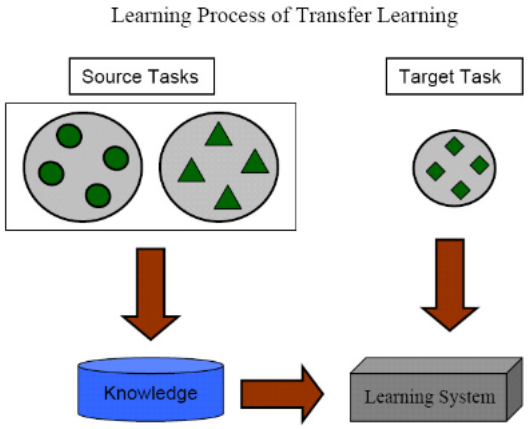


Figure 9: Learning process of transfer learning

2.4.4 Adversarial Training (FGM & FGSM)

Adversarial training is an important way of enhancing the robustness of a neural network. During adversarial training, samples are mixed with small perturbations (changes that are small but likely to cause misclassification) and the network is then adapted to be robust to the adversarial samples.

Adversarial training can be summarised in the following maximum minimisation formula.

$$min_{\theta} E_{(Z,y)\sim D}\left[maxL\left(f_{\theta}(X+\sigma)\right)\right]$$

The inner layer (in brackets) is a maximizer, where X denotes the input representation of the sample, σ denotes the perturbation superimposed on the input, f_{θ} is the neural network function, y is the label of the sample, and $L\left(f_{\theta}(X+\sigma)\right)$ denotes the perturbation σ superimposed on the sample X, which is then passed through the neural network function, compared with the loss obtained from the label y. max(L) is the optimization objective, i.e. finding the perturbation that maximizes the loss function, which simply means that the perturbation added should be as confusing as possible for the neural network.

The outer layer is the minimisation formula for optimising the neural network, i.e. when the perturbation is fixed, we train the neural network model to minimise the loss on the training data, i.e. to make the model robust to such perturbations.

The FGM (Fast Gradient Method) and FGSM (Fast Gradient Sign Method) methods were proposed by Miyato, Takeru, et al. (2021) and Goodfellow, Ian J., et al. (2015) respectively. The difference between FGM and FGSM lies in the different normalization methods used, with FGM using L2 normalization and FGSM taking max normalization of the gradient by the Sign function. max normalisation means that if the value of a dimension of the gradient is positive, it is set to 1; if negative, it is set to -1; if 0, it is set to 0. In theory, L2 normalisation preserves the direction of the gradient more strictly, but max normalisation does not necessarily have the same direction as the original gradient.

FGM: $\sigma = \epsilon \cdot Sign(g)$

FGSM: $\sigma = \epsilon \cdot \left(g / ||g|\right|_2)$

Where g $=\nabla_X\left(\mathrm{L}(\mathrm{f}_\theta(\mathrm{X}), \mathrm{y})\right)$, which is the gradient of the loss function L with respect to the input X. This gradient is easy to find out when we do neural network optimisation.

Both methods of course have the assumption that the loss function is linear or at least locally linear. If it is not (locally) linear, then the direction of the gradient boost is not necessarily the optimal direction.

2.4.5 Exponential Moving Average (EMA)

Moving Average

Suppose we are given the values of a parameter a at different epochs $a_1, a_2, \ldots, a_t, a_t$ then the Moving Average at the end of training is

$$mv_t = \text{ decay } * mv_{t-1} + (1 - \text{ decay }) * a_t$$

decay represents the decay rate, which is used to control the rate at which the model is updated. With the above equation, it is easy to obtain

$$mv_t = \sum_{i=1}^{t} decay^{t-i} * (1 - decay) * a_t$$

when $t - i > C$ and C is infinite

$$decay^{t-i} * (1 - \text{ decay }) * a_t \approx 0$$

so

$$mv_t \approx \sum_{i=t-C}^{t} \text{decay }^{t-i} * (1 - \text{ decay }) * a_t$$

i.e. mv_t is only related to $a_{t-c}, a_2, \dots, a_t$ are related.

Introduction of EMA

Having set the scene before, the formula for EMA is introduced below.

shadowVariable = decay $*$ shadowVariable + (1 - decay) $*$ Variable

shadowVariable is the final parameter value after the EMA process, and Variable is the parameter value of the current epoch round.

EMA maintains a shadow variable for each variable to be updated for training learning. The initial value of the shadow variable is the initial value of this variable.

From the above equation, the decay controls the rate at which the model is updated, with the larger the decay the more stable it becomes. In practice, this is usually set to a constant very close to 1 (0.999 or 0.9999).

2.4.6 Label Smoothing

Problems with Cross-entropy

In a multi-categorization task, the neural network outputs a confidence score for the current data corresponding to each category, and normalizes these scores by softmax, which will eventually give the probability that the current data belongs to each category.

$$q_i = \frac{exp(z_i)}{\sum_{k=1}^{K} exp(_j)}$$

The cross-entropy loss function is then calculated as follows.

$$Loss = -\sum_{i=1}^{K} p_i \, log(q_i)$$

where p_i is the 0-1 hard one-hot label. When training the neural network, the cross-entropy between the predicted probability and the true probability of the label is minimized to obtain the optimal predicted probability distribution. The optimal prediction probability distribution is.

$$z_i = \begin{cases} +\infty, & if(i == y) \\ 0, & if(i \neq y) \end{cases}$$

The following problems exist with cross-entropy.

- The neural network drives itself to learn in the direction with the largest difference between correct and incorrect labels, which can lead to overfitting of the network when the training data is small and insufficient to characterise all the sample features.
- There may be instances of mislabelling in the labeled data, leading the model to trust the correct label too much and ultimately perform poorly on the validation set.

Introduction of Label Smoothing

label smoothing (Müller, Rafael, et al. 2021) can solve the above problem, which is a regularization strategy, mainly by adding noise through soft one-hot, reducing the weight (ϵ) of the category of the true sample labels in the calculation of the loss function, which ultimately has the effect of suppressing overfitting. The true probability distribution after adding label smoothing has been changed as follows.

$$p_i = \begin{cases} 1 - \epsilon, & if(i == y) \\ \dfrac{\epsilon}{\{K-1\}}, & if(i \neq y) \end{cases}$$

$$Loss_i = \begin{cases} (1 - \epsilon) * Loss, & if(i == y) \\ \epsilon * Loss, & if(i \neq y \end{cases}$$

2.4.7 Data Augmentation

In NLP, methods for performing data augmentation can manipulate text, such as replacing tokens, breaking up parts of the original text with information, etc. (Chaudhary, Amit. 2020). In subsequent pages, it may be abbreviated to AUG.

2.5 Related Work

In everyday life, lying is widely recognized for its pervasiveness, with one research suggesting that 40% of adults tell a lie at least once a day (Serota, Levine, & Boster. 2010). People's capacity to detect lying, on the other hand, is not as good as the frequency of lies.

2.5.1 Detecting Deception from Linguistic Styles

Introduction

In order to improve the ability of individuals to identify lies and avoid the negative effects of collection, psychologists have carried out in-depth research on deceptive clue recognition. It mainly focuses on three aspects: (1) Nonverbal intelligence, which recognizes clues through the characteristic actions of the liar, celebrity changes, tone of voice, speaking speed, etc. (Sporer & Schwandt. 2007; Warren, Schertler, & Bull. 2009); (2) Physiological wisdom, which provides judgment by recording the liar's physiological response through a biofeedback device. (Zhang Tingyu, Zhang Yuqing. 2008); (3) Wisdom in language content, judging content analysis through the substance of lies, including standard-based analysis and real monitoring (Vrij, Edward

Roberts, Bull. 2000). The lower correct rate is because they understand the content wisdom of the language and use two kinds of wisdom, real or wrong. For example, actions and characters are often regarded as important clues by us, but a lot of research has found this There is no significant correlation between the two nonverbal behaviors and clues (Reinhard & Sporer. 2008).

Nonetheless, a lot of researchers feel that utilizing linguistic content cues leads to a higher rate of lie recognition than using nonverbal cues. The accuracy percentage of participants in detecting lies will approach 70%-75% when applying linguistic content cues alone (Fuller, Biros, & Delen. 2011).

There are several features of linguistic style, such as pronoun use, emotionally toned words, and prepositions and conjunctions that signal cognitive work, have been linked to a number of behavioral and emotional outcomes. (Newman ML, Pennebaker JW et al. 2003)

According to Newman ML (2003)'s study, at least three language aspects should be linked with deception: (a) fewer self-references (b) more negative emotion terms, and (c) fewer cognitive complexity markers.

Firstly, deceivers use fewer first-person singular pronouns (e.g., I, me, and my) in that liars avoid assertions of ownership either to disconnect themselves from their words or owing to a lack of personal experience (Buller et al. 1996). Secondly, there are more negative emotional words (e.g., hate, worthless, sad) in deceptive communications because deceivers may feel guilty and uncomfortable when lying (DePaulo et al. 2003). Finally, liars use fewer exclusive words (e.g., except, but, without) because creating a false story needs more cognitive resources, leading liars to tell fewer complex stories (cf. Richards & Gross. 1999, 2000).

Methodology

The methodology used in the present research allowed us to test whether differential use of multiple cues (e.g., self-references + negative words + cognitive words) is a reliable marker of deception. (Newman ML, Pennebaker JW et al. 2003).

To increase the generalizability of our language data, they asked participants to either lie or tell the truth about various topics in various circumstances. Each of the five studies' basic methodology is outlined below. The LIWC programme was used to examine each participant's written or transcribed spoken samples in all of the investigations. The five studies are Videotaped abortion attitudes, typed abortion attitudes, handwritten abortion attitudes, feelings about friends, and mock crime.

As a result, the current studies suggest that liars may be reliably detected by their words—not what they say, but how they say it.

2.5.2 Detecting Deception by Text Classification

Introduction

Text classification is the process of categorizing the text into a group of words. By using NLP (Natural Language Processing), which refers to a software system developed by integrating linguistic theory and computer technology that can quantitatively analyse text content (Duran, Hall, Mccarthy, & Mcnamara, 2010), text classification can automatically analyse text and then assign a set of predefined tags or categories based on its context. Although NLP technology is still immature and in the development stage, it has achieved certain results. The most commonly used software system is language acquisition and word count (Linguistic Inquiry and Word Count, LIWC) (Pennebaker, Francis, & Booth, 2001). LIWC can automatically analyse the text in terms of words, compare all words in the text with norm words, classify them into 72 dimensions, and calculate the proportion of words in each dimension to the total number of words. Newman et. al (2003) has used it in the field of lie recognition and found it to be effective.

Methodology

To implement deception detection using classification, a lot of researchers have used machine learning classifiers, such as naïve Bayes (NB), support vector machine (SVM), decision tree, logistic regression (LR) and neural networks. The general classification process comprises dataset selection, data pre-processing (tokenizing, stemming, etc.), features extractions, classifier training, and content classification.

For example, Ahmed et al. (2017) investigated 2 different features extraction techniques including TF and TF-IDF and 6 different machine learning classification algorithms, namely, SVM, linear support vector machines (LSVM), K-nearest neighbour (KNN), LR, and decision trees (D. The proposed model was evaluated using 3 different datasets, involving truthful and fake content (both news articles and reviews). The experiments started with unigram (n=1) to Four-gram (n=4), and all used 5-fold cross validation.

Finally, the result indicated that the linear SVM can reach 90% accuracy. It's also apparent that they obtain better accuracy with higher feature values, since they employ 10,000 and 50,000 feature values, respectively. We can also see that increasing the n-gram size reduces accuracy when all classifiers are used. Because false and honest evaluations featured comparable terms, TF-IDF performed better than TF. KNN with 4-gram words and 50 000 feature values yielded the lowest accuracy, less than random probability.

Another group of researchers (Mihalcea and Strapparava, 2009) did the experiments using two classifiers: Naïve Bayes and SVM working on three different datasets labelled with truth value explicitly: opinions on abortion, opinions on death penalty, and feelings about the best friend. The pre-processing only included tokenization and stemming and there was no feature selection and stop-word removal.

The average classification performance of 70% – substantially better than the baseline of 50% – showed that good separation is possible. Also, they concluded more data can improve performance and it relies on clues specific to truth/deception and is not topic specific.

Moreover, introducing the dominance score helped figure out how dominant the classes from LIWC are. Specifically, top 5 classes of deceptive text are METAPH, YOU, OTHER, HUMANS, and CERTAIN whilst top 5 classes of truthful text are OPTIM, I, FRIENDS, SELF, and INSIGHT respectively.

2.5.3 Transfer Learning

(1) Transductive Transfer Learning

Definition:

Transductive transfer learning aims to improve the learning of the target predictive function fT () in DT using the knowledge in DS and TS, where DS = DT and TS = TT, given a source domain DS and a corresponding learning task TS, a target domain DT and a corresponding learning task TT. Furthermore, some unlabelled target domain data must be available during training (Arnold et al.).

(a) Domain Adaptation

Definition:

Domain-adaptation refers to the process of adapting to a new domain. This is common when we want to learn a new data distribution in the target domain.

Detailed Literature Review:

Ruder et al. (2017) applied a teacher-student model for transferring knowledge from multiple domains to a single domain in an unsupervised approach.

Shah et al. (2018) applied adversarial domain adaptation for the detection of duplicate questions. Their approach consists of three components where the first component encodes the question.

Ma et al. (2019) considered the task of domain shift from pretrained BERT to other target domains

(b) Cross-lingual Learning

Definition:

Cross-lingual Learning is the process of adapting to a different language in the target domain. This is common when we want to use a high-resource language to learn tasks in a low-resource language. Cross-lingual language modelling, for example, has been investigated to understand how it affects low-resource languages (Adams et al.2017).

Detailed Literature Review:

Kim et al. (2017) invented a model for pos-tagging in a cross-lingual context with varied input sizes for the input and output languages.

Schuster et al. (2018) proposed a new dataset of 57k annotated utterances in English, Spanish, and Thai classified as weather, alert, and reminder.

(2) Inductive Transfer Learning

Definition:

Given a source domain DS and a learning task TS, a target domain DT and a learning task TT, inductive transfer learning tries to aid in the learning of the target predictive function fT () in DT using information from DS and TS, where TS ≠ TT (S. J. Pan, 2009).

(a) Sequential Transfer Learning

Definition:

Sequential transfer learning refers to the process of learning multiple tasks in a sequential fashion. For example, given a pretrained model M, we would want to transfer the learning to several tasks (T1, T2, Tn). We learn a specified task Tt at each time step t. It is slower than multi-task learning, although it might be useful when not all tasks are available at the moment of training. Sequential transfer learning is further classified into four types, which are fine-tuning, adapter modules, feature based, and zero-shot individually.

Detailed Literature Review:

Lee et al. (2017) tested transfer learning performance on the patient note de-identification task. This challenge is a version of named entity recognition in which the model must identify sensitive information about the patient in order to deidentify.

McCann et al. (2017) investigated the transferability of word vectors learned on different-sized machine translation datasets.

Lakew et al. (2018) advocated employing a shared dynamic vocabulary to transmit information across neural machine translation models.

Howard and Ruder. (2018) suggested ULMFiT, a method to universal fine-tuning for text categorization. They use a similar approach to the computer vision transfer learning problem in that a model is trained on ImageNet and then fine-tuned at the end by adding classification layers.

(b) Multi-Tasks Learning

Definition:

Multi-Tasks Learning refers to the process of learning multiple tasks at the same time. All tasks are learned concurrently. It might be claimed that learning several tasks from a single dataset is preferable to learning tasks separately (Evgeniou and Pontil, 2004).

Detailed literature review:

Liu et al. (2019) investigated multitask deep neural networks (MT-DNN). The training process is divided into two sections. The first stage involves training a language understanding model with a BERT model. The second step includes the addition of four models for task-specific adaptation.

Raffel et al. (2019) investigated the impact of employing a unified text-to-text transfer transformer (T5). They use an encoder-decoder network, similar to the Transformers paradigm (Vaswani et al. 2017).

Table 1: Summary of literature categorized in pretrained models used, tasks tackled and type of transfer learning approach.

Reference	Base Model	NLP Tasks	Transfer Learning
Mou et al. 2016	LSTM-RNN Siamese CNN	Sentiment Analysis Sentence Pair Classification	Fine Tuning Features Based
Peters et al. 2017	CNN-LSTM	Named Entity Recognition	Features Based
Lee et al. 2017	LSTM	Named Entity Recognition	Fine Tuning
Ruder et al. 2017	MLP	Sentiment analysis	Domain Adaptation
McCann et al. 2017	OpenNMT	Sentiment analysis Entailment Question Classification Question Answering	Fine Tuning
Kim et al. 2017	BLSTM	Part of Speech Tagging	Cross-lingual Fine Tuning
Schuster et al. 2018	ELMoCoVe	User Intent Classification	Cross-lingual Fine Tuning
Lakew et al. 2018	OpenNMT	Translation	Fine Tuning Domain Adaptation
Howard and Ruder. 2018	AWD-LSTM	Text Classification	Discriminative Fine Tuning
Devlin et al. 2018	BERT	Language understanding Natural Language Inference Question Answering	Fine Tuning Features Based
Peters et al. 2018	ELMo	Question Answering Text Entailment Semantic Role Labelling Coreference Resolution Named Entity Extraction Sentiment Analysis	Fine Tuning
Shah et al. 2018	BLSTM	Duplicate Question Detection	Domain Adaptation
Yin et al. 2019	BERT	Topic Categorization Emotion Detection	Zero-shot

		Situation Frame Detection	
Stickland and Murray. 2019	BERT	Natural Language Understanding	Adapter Modules
Ma et al. 2019	BERT	Natural Language Inference Answer Sentence Selection Paraphrase Detection	Domain Adaption
Houlsby et al. 2019	BERT	Text Classification	Adapter Modules
Semnani and Sadagopan.2019	BERT	Question Answering	Adapter Modules
Radford et al. 2019	GPT-2	Language Modeling Children Book Test Long Range Dependencies Reading Comprehension Summarization Translation Question Answering	Zero-shot Transfer
Liu et al. 2019b	RoBERTa	Question Answering Natural Language Understanding Reading Comprehension	Fine Tuning
Liu et al. 2019a	BERT	Natural Language Understanding	Multi-task Fine Tuning
Yang et al. 2019	XLNet	Natural Language Inference Question Answering Sentiment Analysis Document Ranking	Fine Tuning

Table 1: Summary of literature of transfer learning

3. Descriptive Analysis

In this project, the task was to recognize whether a piece of text is deceptive or truthful. In order to fulfil our goals, collecting proper datasets was one of the most critical parts of the research. Even if textual lies are widespread on the Internet, it was challenging to find labelled datasets that met our requirements. We utilized two datasets, one is fake news dataset for regular training process, and the other is US English cross culture dataset used for transfer learning. Next, we will describe these two datasets separately

3.1 Fake News Dataset

3.1.1 Data Gathering

The fake news dataset includes general fake news sub-dataset and celebrity news sub-dataset.

The general fake news sub-dataset is collected by Veronica et al. (2017), which contains two subsets, fake news dataset and celebrity news dataset. The fake news dataset contains news in six different domains: technology, education, business, sports, politics, and entertainment. The real news in the dataset was gathered from a range of popular news sources, mostly in the United States, including ABCNews, CNN, USAToday, NewYorkTimes, FoxNews, Bloomberg, and CNET, among others. The false news in this dataset was made up of fake copies of the real news in the dataset that were generated using Mechanical Turk.

The Celebrity sub-dataset collected by Baltimore, Maryland. (2014) contains news about celebrities (actors, singers, socialites, and politicians). The real news in the dataset came from mainstream news websites' entertainment, fashion, and style sections, as well as entertainment magazine websites. The fake news came from gossip websites including Entertainment Weekly, People Magazine, RadarOnline, and other tabloid and entertainment-oriented magazines. The news articles were collected in pairs, with one article being real and the other being a fake (rumors and false reports). The articles were personally validated using gossip-checking websites like "GossipCop.com," and they were also cross-referenced with information from other online entertainment news sources.

3.1.2 Description

In general, the dataset contains 938 pieces of text, having two themes, news and celebrity, which are 455 and 483 respectively, as displayed in Figure 10. It is divided into training set and test set. The training set occupies 75%, which has 703 pieces, whereas the test set remains 25%, which has 235 pieces, which is shown in Figure 11.

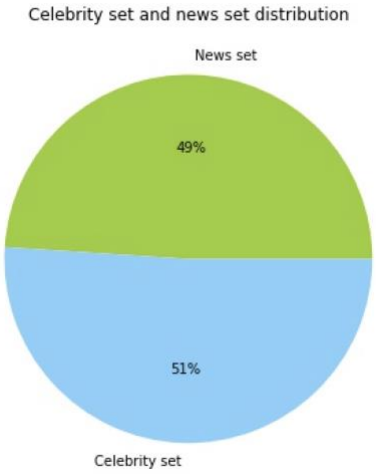


Figure 10: Fake news theme distribution

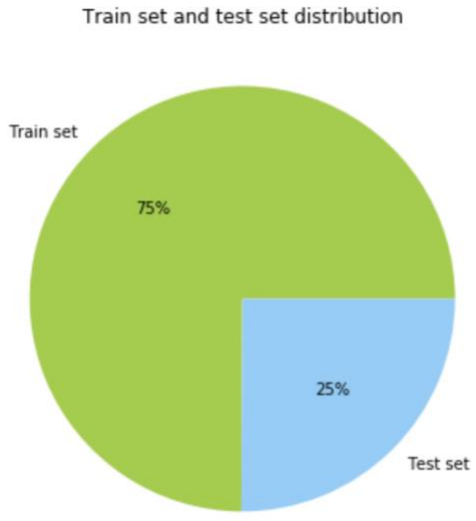


Figure 11: Fake news train set and test set distribution

3.1.3 Data Classification

There are 938 pieces of text of the whole dataset, in which 479 are labelled 0 (truthful), while 459 are labelled 1 (deceptive).

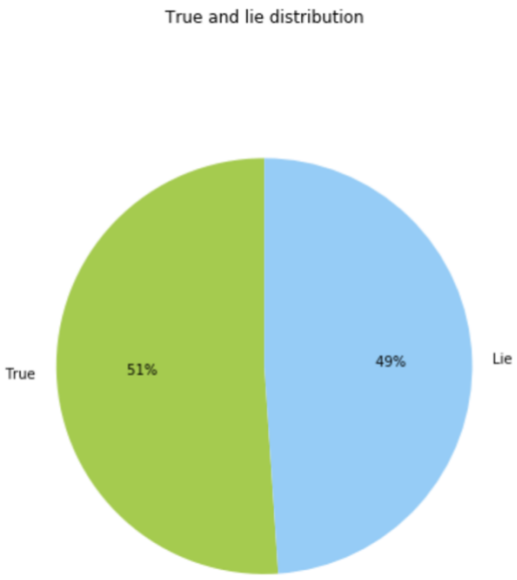


Figure 12: Fake news true and lie distribution

3.1.4 Statistical Text Length

As for the length of text in training set, the result is shown in Figure 13. To be more specific, the average length of 938 pieces of text is about 345 words. The longest piece of text has 1573 words, whilst the shortest one has just 2 words. 75% of the pieces of text are no more than 511 words. These results are visualized in Figure 14.

	text_len
count	938.000000
mean	345.414712
std	259.911174
min	2.000000
25%	152.000000
50%	237.000000
75%	510.750000
max	1573.000000

Figure 13: Fake news text length

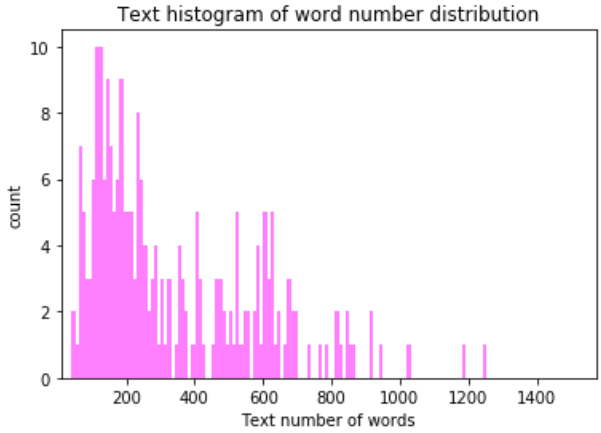


Figure 14: Fake news text length histogram

3.1.5 Word Frequency

Furthermore, the number of times each word appears in the train set has been statistically analyzed. Before counting the word frequency, the useless information such as punctuations, stop words and messy code were filtered. Furthermore, the words have been stemmed, which means that they are formatted to the original formats (For instance, likes, liking, liked to like).

After the simple text processing, it can be known there are 7867 words in text except the meaningless information. The most frequent 20 words is displayed in Table 2 and Figure 15. “Said” appears 261 times, which is the most frequently used word in the training set, followed by “year”, 161 times.

Word	Number of times
said	65
year	52
new	47
school	42
student	40
one	34
last	31
time	27
u	25
would	24
trump	24
president	23
also	22
star	22
child	21
first	21
could	20
week	19
company	19
home	18

Table 2: Fake news top 20 frequent words

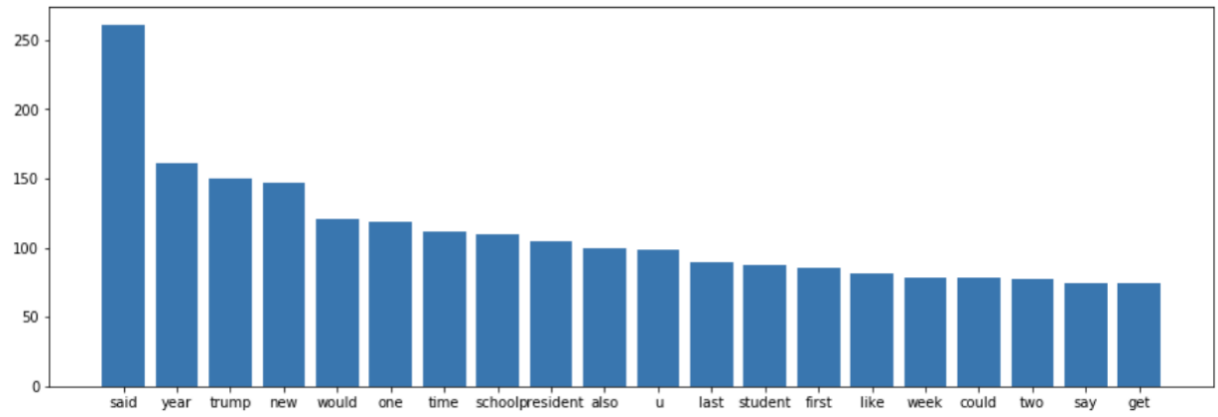


Figure 15: News top 20 frequent words

3.2 Cross-cultural English US Dataset

3.2.1 Data Gathering

Cross-cultural dataset is collected by Veronica et al. (2014). Cross-cultural dataset contains 4 subsets, which are English US, English India, Spanish Mexico, and Romanian respectively. We only used English US dataset because it is the most common used language out of the 4. It was collected from English speakers using Amazon Mechanical Turk.

3.2.2 Description

In general, the dataset contains 458 pieces of text, having 3 themes, abortion, best friend and death penalty, which are 153, 153 and 152 respectively, as displayed in Figure 16. It is divided into training set and test set. The training set takes up 78%, which has 359 pieces, whereas the test set holds 22%, which has 99 pieces, which is shown in Figure 17.

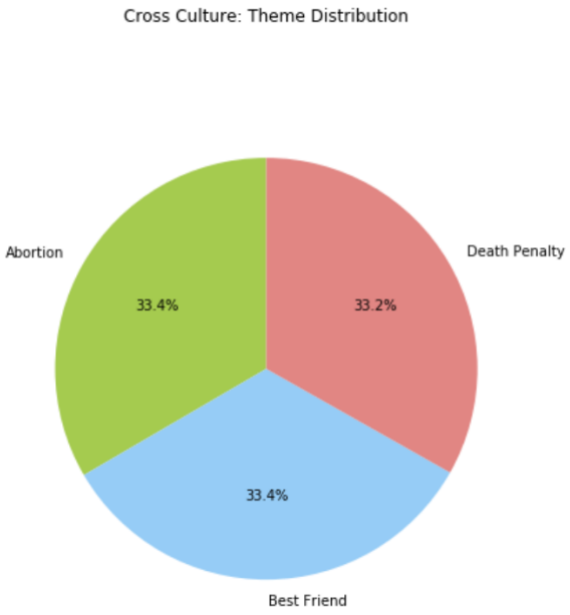


Figure 16: Cross-cultural theme distribution

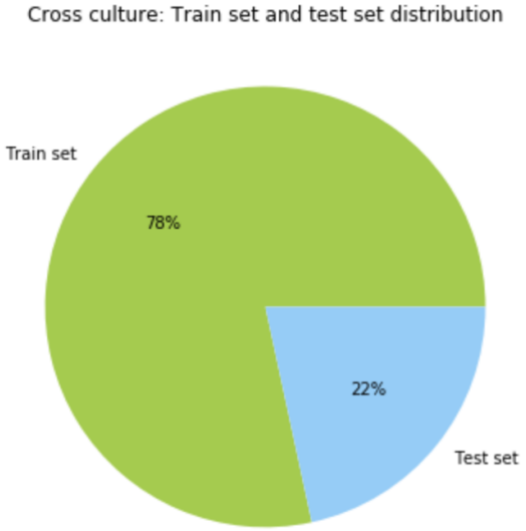


Figure 17: Cross-cultural train set and test set distribution

3.2.3 Data Classification

There are 458 texts of the whole dataset, in which 230 are labelled 0 (true), while 228 are labelled 1 (lie), which are relatively equal, shown in Figure 18.

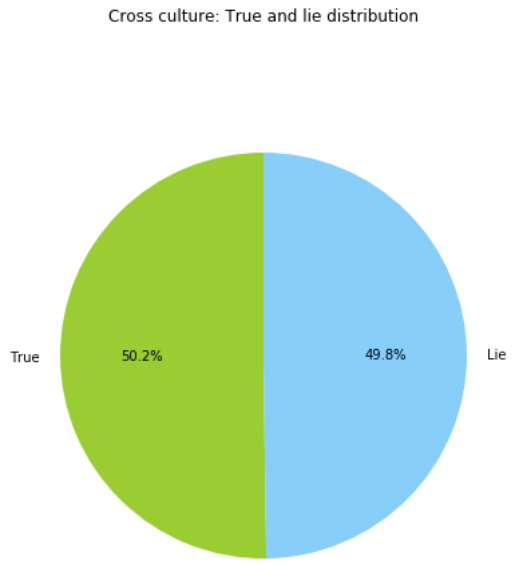


Figure 18: Cross-cultural true and lie distribution

3.2.4 Statistical Text Length

As for the length of text in training set, the result is shown in Figure 19. To be more specific, the average length of 938 texts is about 410 words. The longest piece of text has 1153 words, whilst the shortest one has just 23 words. 75% of the pieces of text are no more than 488 words. These results are visualized in Figure 20.

	text_len
count	458.000000
mean	409.506550
std	180.808544
min	23.000000
25%	293.000000
50%	375.000000
75%	487.750000
max	1153.000000

Figure 19: Cross-cultural text length

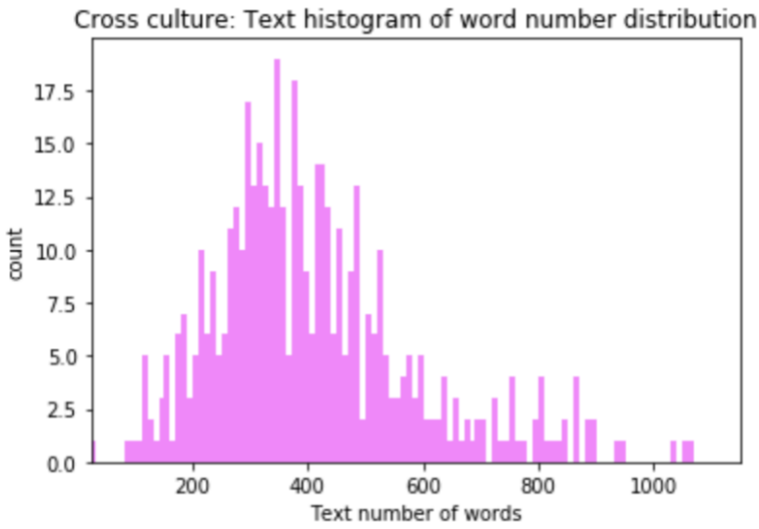


Figure 20: Cross-cultural text length histogram

3.2.5 Word Frequency

Finally, the frequency with which each word appears in the train set has been statistically examined. Before calculating the frequency of words, unnecessary information such as punctuation, stop words, and messy code were removed. Furthermore, the words have been stemmed, which means they have been formatted in accordance with the original formats

After the simple text processing, it can be known there are 2767 words in the dataset. The most frequent 20 words is displayed in Table 3 and Figure 21. The most frequent word is “death” which appears 423 times, the second one is “abortion”, 369 times, and the third frequently used word is life, 321 times, which fit the key words three themes.

Word	Number of times
death	423
abortion	369
penalty	331
life	321
believe	226

woman	219
people	204
right	174
child	173
person	146
crime	146
would	134
think	120
baby	106
criminal	103
one	97
murder	97
get	96
want	93
someone	92

Table 3: Fake news top 20 frequent words

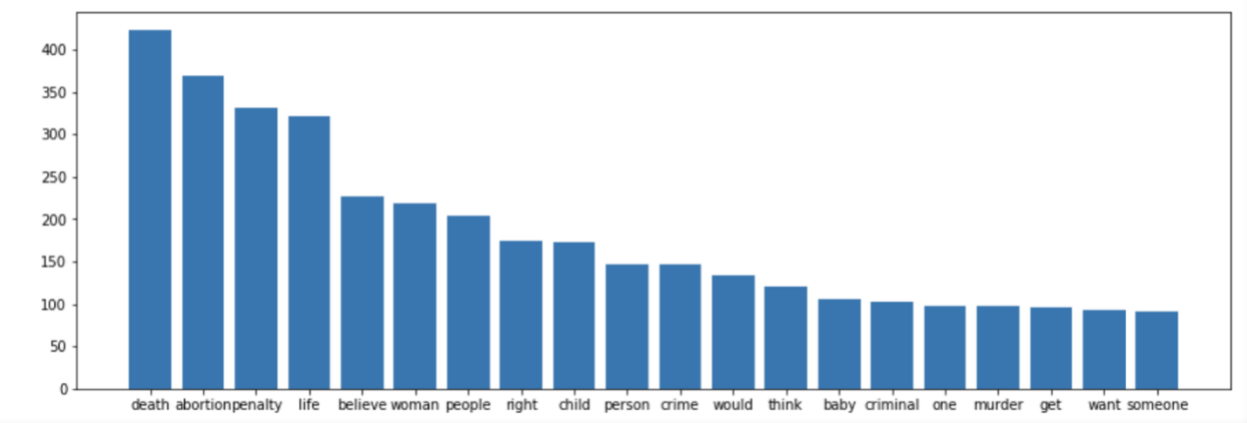


Figure 21: Cross-cultural top 20 frequent words

4. Implementation

After data collection and exploratory analysis, we built several supervised binary classification models based on bag-of-words on word count feature, bag-of-words on TF-IDF feature, Part-of-

speech feature, and sentiment lexicon feature to differentiate between true and false texts from the fake news dataset.

To begin, seven fundamental models were created using scikit-learn, including Naive Bayes, logistic regression, SVM, decision tree, random forest, and K-nearest neighbours algorithms. Following that, two neural networks - LSTM and BERT were processed. Finally, we utilised transfer learning to generalise the pre-trained weights we had learned on the cross-cultural English US dataset.

4.1 Pre-processing

In that each piece of texts of raw data were stored in separate fake and legit files with no labels, we firstly concatenated those files into one .csv file with labels, 0 is true (legit), 1 is false (fake). Then we split the data into training dataset and test dataset by scikit-learn, of which train dataset occupied 75%, whereas test dataset took up 25%. The train dataset was used for feature extraction and model training, while test dataset was used for unbiased evaluation. The processing of cross-cultural English US dataset was much at one.

Special characters and punctuation were typically treated as insignificant and uninformative portions of text in standard word-level pre-processing. However, for online texts such as news, punctuation was deemed non-trivial, and there were few emoticons in the dataset. As a result, the primary goal of pre-processing was to clear the meaningless and noisy words while detecting and maintaining the featured texts as much as possible for feature engineering. With this in mind, we did pre-processing, which was followed by text cleaning, tokenisation, POS tagging, and lemmatisation.

First, we removed any content that did not contribute to the meaning of the text, like general stop words, special characters such as '(),' whitespaces, and so on. We specifically reserved a set of punctuation marks such as commas, full stops, ellipses, question marks, and exclamation points. Tweet NLP POS tagger, a particularly built POS tagger for Twitter data presented by

Gimpel et al. (2011), provided 25 types of tags for POS tagging. Finally, the processed tweets were lemmatised using NLTK's WordNet lemmatiser, which enabled us to extract the root forms of derived words.

4.2 Feature Extraction

As aforementioned, we tackled the given deception classification challenge using both classic machine learning and deep learning approaches. In the first technique, the selection of features was critical for classifier training. We chose four different types of fine-grained feature extraction methods proposed by Van Hee et al. (2018), all of which demonstrated good performance in representing short texts, including bag-of-words features (based on word count and TF-IDF), syntactic POS features, and sentiment lexicon features.

First, while building the corpus's vocabulary, we kept all highlighted texts because of their utility in terms of lie detection. To reduce the impact of both-class-frequent words, the uni-grams vocabulary was constructed after scaling the word frequency by TF-IDF, and we also utilised word count as the foundation for uni-grams as a contrast.

In terms of syntactic POS features, we extracted features for each text the averaged frequency (frequency divided by length of tweet) of the 26 tags, which included not only common POS categories. POS features are considered to be useful since it reflects the intensity of interjection and sentiment, which signals the sign of deception (Kunneman et al. 2015).

Sentiment lexicon features were added based on the existing sentiment lexicon: AFINN (Nielsen et al. 2011), which is dedicated to words. We integrated the lexicons mentioned above and extracted the five numeric and one binary feature shown below for each text:

- the averaged frequency of positive, negative, and neutral lexicon words
- sum of the sentiment score, i.e., overall polarity
- difference between the highest and the lowest sentiment score

• a binary feature that indicate the presence of sentiment contrast (at least one positive and negative pair)

By extracting sentiment features, we could capture the polarity clash, which could explicitly indicate the sense of verbal deception.

4.3 Modelling

4.3.1 Basic Models

In accordance with Van Hee et al. (2018a), combining feature types may improve classification performance. We trained each of 6 classifiers based on 4 individual features and 11 combinations by using functions from scikit-learn library. Consequently, there were 80 implementations of classifiers generated.

4.3.2 Neural Network Models

Following we trained two neural network models, which are LSTM and BERT.

The dataset is about fake news detection. In essence, it can be regarded as a binary classification problem. A label of 1 means fake news, and a label of 0 means not.

(1) Model Structure

In response to this problem, I used the following two model structures. On the left, LSTM is used as a feature extractor, and on the right, a BERT model is used as a feature extractor.

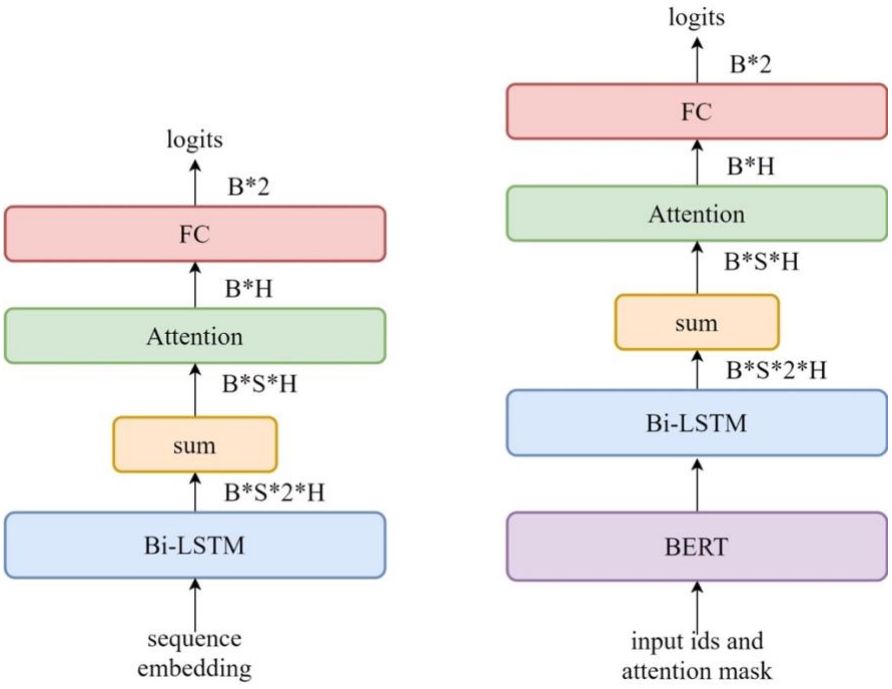


Figure 22: Two neural networks' model structure

(2) Experiment

- Dataset

The fake news dataset was used here. The fake news dataset has a total of 938 pieces of data. 703 pieces of data were used as the training set and the other 235 pieces were used as the test set.

- Parameter Setting

	epoch	lr	batch_size	embed_dim
LSTM	40	5e-4	12	300
BERT	15	1e-5	6	768

Table 4: Parameter setting of LSTM and BERT

Note: The reason as to the difference in embed_dim between the LSTM model and the BERT model was that LSTM performs worse at embed_dim = 768 than at embed_dim = 300, so it was set to 300 dimensions here instead of 768.

- Training Process

For LSTM, Adam (Diederik P. et al) optimizer was used for training, whereas for BERT, AdamW optimizer is applied for training. Both models used the F1-score on the test dataset as the iteration stop condition. Training would stop when there was no rise for 2 consecutive epochs.

To enhance the robust and performance of models, the loss function of label smoothing was used for training, after which the two models are augmented by using different methods, including data augmentation (AUG), EMA, FGM and the combinations among these three methods.

During the training process, the loss and accuracy change curves of the training processes of different augmented methods and combinations of two models were drawn, as shown in the figures below.

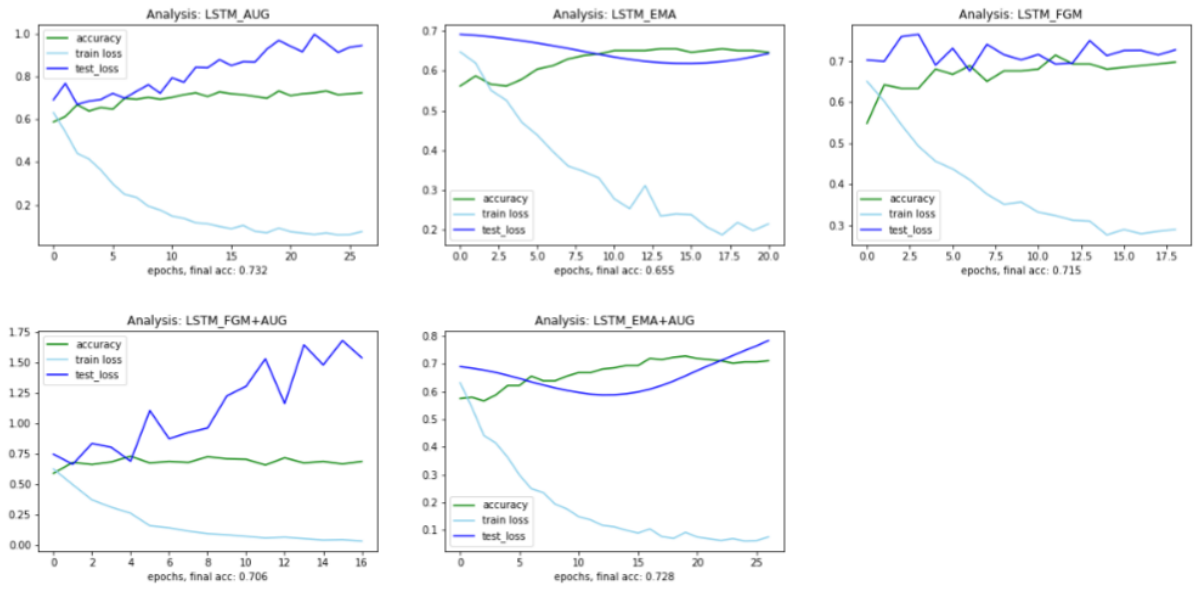


Figure 23: LSTM training metrics

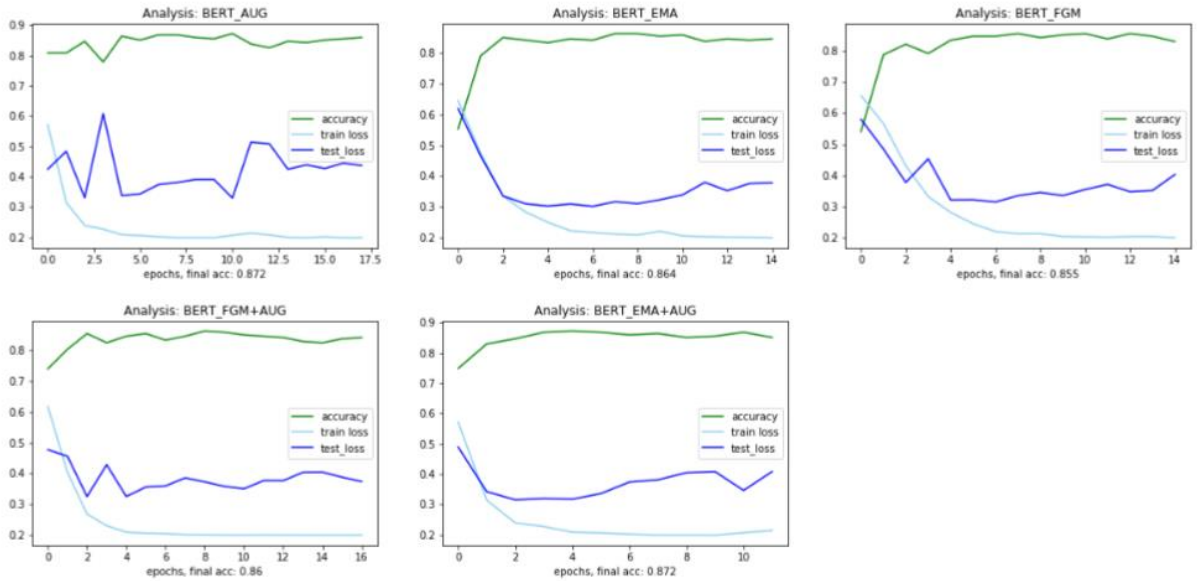


Figure 24: BERT training metrics

4.3.3 Transfer Learning

This part used the idea of transfer learning to verify the transfer learning ability of the BERT model. We applied a new BERT model training on the cross-cultural English US dataset based on the pre-trained weights from the previous BERT model above.

- Dataset

The cross-cultural English US dataset was used here. This dataset contains 458 pieces of data, comprising 3 themes – abortion, best friend, and death penalty, which are 153, 153, and 152 separately. 359 pieces of data were used as training set whilst 99 were for test set.

- Parameter Setting

	epoch	lr	batch_size	embed_dim
BERT	15	1e-5	6	768

Table 5: Parameter setting of transfer learning

- Training Process

Same as the LSTM model and the former BERT model, label smoothing, data augmentation (AUG), EMA, FGM were used in the new BERT model.

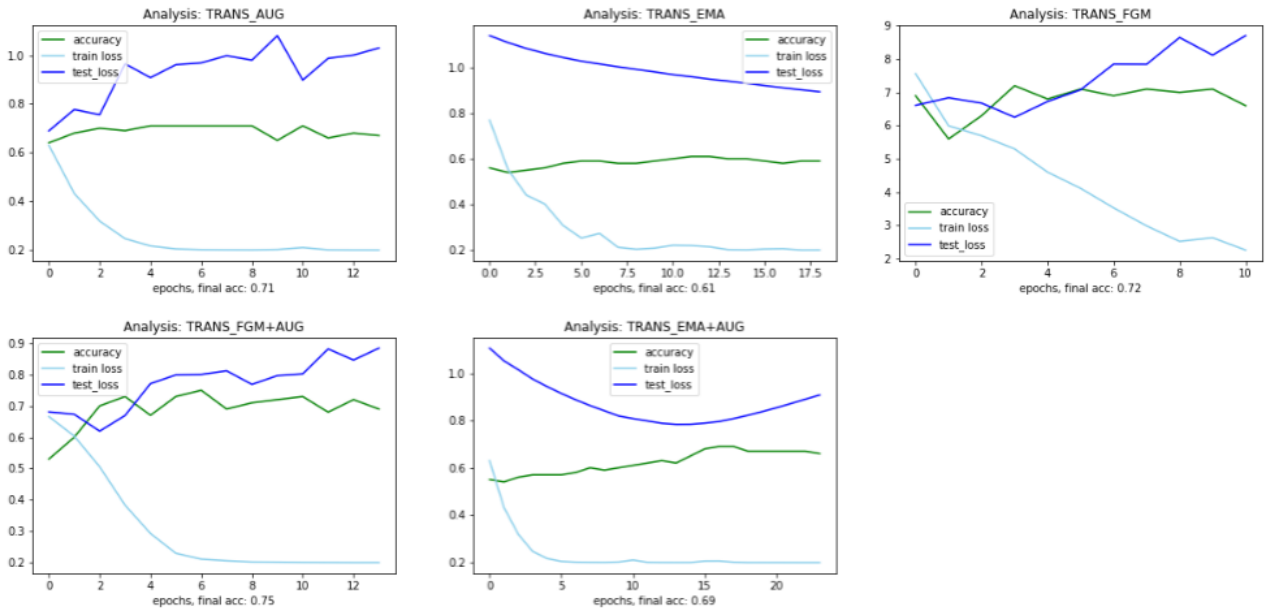


Figure 25: BERT for transfer learning training metrics

5. Results and Evaluations

Note: Statistics are rounded to two decimal places.

Classifier	Features	True class (0)			Lie class (1)			Overall			
		Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Acc
Naive Bayes	Word count	0.74	0.62	0.68	0.71	0.81	0.76	0.73	0.72	0.72	0.72
	Bag-of-words on TF-IDF	0.68	0.83	0.75	0.82	0.67	0.73	0.75	0.75	0.74	0.74
	POS	0.60	0.69	0.64	0.69	0.60	0.64	0.64	0.64	0.64	0.64
	Sentiment	0.53	0.76	0.63	0.67	0.42	0.52	0.60	0.59	0.57	0.58
	Count, bag-of-words	0.84	0.61	0.70	0.72	0.90	0.80	0.78	0.75	0.75	0.76
	Count, POS	0.77	0.61	0.68	0.72	0.84	0.77	0.74	0.73	0.73	0.74
	Count, sentiment	0.62	0.62	0.62	0.67	0.67	0.67	0.65	0.65	0.65	0.65

	BOW, POS	0.89	0.30	0.45	0.62	0.97	0.75	0.75	0.64	0.60	0.66	
	BOW, sentiment	0.59	0.56	0.58	0.64	0.67	0.65	0.61	0.61	0.61	0.62	
	POS, sentiment	0.56	0.80	0.66	0.73	0.47	0.57	0.65	0.63	0.62	0.62	
	Count, BOW, POS	0.86	0.57	0.69	0.71	0.92	0.80	0.79	0.74	0.72	0.76	
	Count, BOW, sentiment	0.60	0.57	0.58	0.64	0.67	0.65	0.62	0.62	0.62	0.62	
	Count, POS, sentiment	0.62	0.62	0.62	0.67	0.67	0.67	0.65	0.65	0.65	0.65	
	BOW, POS, sentiment	0.59	0.56	0.58	0.64	0.67	0.65	0.61	0.61	0.61	0.62	
	All features	0.60	0.57	0.58	0.64	0.67	0.65	0.62	0.62	0.62	0.62	
Linear Regression	Word count	0.74	0.74	0.74	0.78	0.78	0.78	0.76	0.76	0.76	0.76	Baseline
	Bag-of-words on TF-IDF	0.72	0.81	0.76	0.77	0.77	0.77	0.77	0.77	0.77	0.77	
	POS	0.56	0.73	0.63	0.68	0.49	0.57	0.62	0.61	0.60	0.60	
	Sentiment	0.54	0.70	0.61	0.65	0.48	0.55	0.59	0.59	0.58	0.58	
	Count, bag-of-words	0.74	0.74	0.74	0.78	0.77	0.77	0.76	0.76	0.76	0.76	
	Count, POS	0.74	0.76	0.75	0.79	0.77	0.78	0.77	0.77	0.77	0.77	
	Count, sentiment	0.75	0.75	0.75	0.78	0.78	0.78	0.76	0.77	0.76	0.77	
	BOW, POS	0.64	0.80	0.71	0.78	0.62	0.69	0.71	0.71	0.70	0.70	
	BOW, sentiment	0.65	0.78	0.71	0.77	0.63	0.70	0.71	0.71	0.70	0.70	
	POS, sentiment	0.62	0.76	0.68	0.74	0.60	0.66	0.68	0.68	0.67	0.67	
	Count, BOW, POS	0.75	0.77	0.76	0.80	0.78	0.79	0.77	0.77	0.77	0.77	
	Count, BOW, sentiment	0.75	0.74	0.75	0.78	0.79	0.78	0.76	0.76	0.76	0.77	
	Count, POS, sentiment	0.71	0.71	0.71	0.75	0.75	0.75	0.73	0.73	0.73	0.73	
	BOW, POS, sentiment	0.59	0.70	0.64	0.69	0.58	0.63	0.64	0.64	0.63	0.63	
	All features	0.71	0.68	0.69	0.73	0.76	0.75	0.72	0.72	0.72	0.72	

SVM	Word count	0.62	0.85	0.72	0.81	0.56	0.66	0.72	0.70	0.69	0.69
	Bag-of-words on TF-IDF	0.69	0.81	0.75	0.81	0.69	0.74	0.75	0.75	0.74	0.74
	POS	0.56	0.85	0.68	0.77	0.43	0.55	0.67	0.64	0.61	0.63
	Sentiment	0.55	0.83	0.66	0.74	0.41	0.53	0.65	0.62	0.60	0.61
	Count, bag-of-words	0.63	0.85	0.72	0.82	0.56	0.67	0.72	0.71	0.70	0.70
	Count, POS	0.58	0.86	0.70	0.80	0.47	0.59	0.69	0.67	0.64	0.65
	Count, sentiment	0.56	0.83	0.67	0.75	0.44	0.55	0.66	0.64	0.61	0.62
	BOW, POS	0.56	0.85	0.68	0.77	0.43	0.55	0.67	0.64	0.61	0.63
	BOW, sentiment	0.55	0.83	0.67	0.75	0.42	0.54	0.65	0.63	0.60	0.61
	POS, sentiment	0.57	0.85	0.68	0.78	0.44	0.57	0.67	0.65	0.62	0.63
	Count, BOW, POS	0.58	0.86	0.70	0.80	0.47	0.59	0.69	0.67	0.64	0.65
	Count, BOW, sentiment	0.56	0.83	0.67	0.75	0.44	0.55	0.66	0.64	0.61	0.62
	Count, POS, sentiment	0.58	0.85	0.69	0.78	0.46	0.58	0.68	0.66	0.63	0.64
	BOW, POS, sentiment	0.57	0.85	0.68	0.78	0.44	0.57	0.67	0.65	0.62	0.63
	All features	0.58	0.85	0.69	0.78	0.46	0.58	0.68	0.66	0.63	0.64
Decision Tree	Word count	0.66	0.71	0.68	0.73	0.68	0.79	0.69	0.69	0.69	0.69
	Bag-of-words on TF-IDF	0.58	0.79	0.67	0.74	0.51	0.60	0.66	0.65	0.64	0.64
	POS	0.52	0.51	0.52	0.59	0.60	0.59	0.55	0.55	0.55	0.56
	Sentiment	0.51	0.58	0.54	0.59	0.52	0.55	0.55	0.55	0.55	0.55
	Count, bag-of-words	0.61	0.69	0.65	0.70	0.63	0.66	0.66	0.66	0.66	0.66
	Count, POS	0.68	0.72	0.70	0.75	0.71	0.73	0.71	0.72	0.71	0.71
	Count, sentiment	0.63	0.72	0.68	0.73	0.63	0.68	0.68	0.68	0.68	0.68
	BOW, POS	0.64	0.69	0.66	0.71	0.67	0.69	0.68	0.68	0.68	0.68
	BOW, sentiment	0.64	0.79	0.70	0.77	0.61	0.68	0.70	0.70	0.69	0.69
	POS, sentiment	0.51	0.59	0.54	0.59	0.51	0.54	0.55	0.55	0.54	0.54
	Count, BOW, POS	0.69	0.71	0.70	0.74	0.72	0.73	0.71	0.71	0.71	0.71
	Count, BOW, sentiment	0.66	0.76	0.71	0.76	0.66	0.71	0.71	0.71	0.71	0.71

	Count, POS, sentiment	0.69	0.69	0.69	0.73	0.74	0.74	0.71	0.71	0.71	0.71
	BOW, POS, sentiment	0.58	0.62	0.60	0.65	0.61	0.63	0.62	0.62	0.62	0.62
	All features	0.62	0.68	0.65	0.70	0.63	0.66	0.66	0.66	0.66	0.66
Random forest	Word count	0.71	0.83	0.76	0.82	0.71	0.76	0.77	0.77	0.76	0.76
	Bag-of-words on TF-IDF	0.68	0.81	0.74	0.80	0.67	0.73	0.74	0.74	0.74	0.74
	POS	0.58	0.65	0.61	0.66	0.59	0.62	0.62	0.62	0.62	0.62
	Sentiment	0.54	0.62	0.58	0.62	0.53	0.57	0.58	0.58	0.57	0.57
	Count, bag-of-words	0.71	0.82	0.76	0.82	0.71	0.76	0.76	0.76	0.76	0.76
	Count, POS	0.68	0.83	0.75	0.82	0.67	0.73	0.75	0.75	0.74	0.74
	Count, sentiment	0.72	0.83	0.77	0.83	0.72	0.77	0.77	0.77	0.77	0.77
	BOW, POS	0.67	0.81	0.73	0.80	0.66	0.72	0.73	0.73	0.73	0.73
	BOW, sentiment	0.64	0.83	0.72	0.80	0.59	0.68	0.72	0.71	0.70	0.70
	POS, sentiment	0.56	0.70	0.62	0.67	0.53	0.59	0.62	0.61	0.61	0.61
	Count, BOW, POS	0.70	0.82	0.75	0.81	0.70	0.75	0.76	0.76	0.75	0.75
	Count, BOW, sentiment	0.71	0.83	0.76	0.82	0.71	0.76	0.77	0.77	0.76	0.76
	Count, POS, sentiment	0.66	0.82	0.73	0.80	0.63	0.71	0.73	0.73	0.72	0.72
	BOW, POS, sentiment	0.67	0.82	0.74	0.80	0.65	0.72	0.74	0.73	0.73	0.73
	All features	0.73	0.83	0.78	0.84	0.73	0.78	0.78	0.78	0.78	0.78
KNN	Word count	0.47	0.60	0.52	0.54	0.41	0.47	0.50	0.50	0.50	0.50
	Bag-of-words on TF-IDF	0.73	0.54	0.62	0.68	0.83	0.74	0.70	0.68	0.68	0.69
	POS	0.59	0.65	0.62	0.67	0.61	0.64	0.63	0.63	0.63	0.63
	Sentiment	0.55	0.65	0.60	0.64	0.55	0.59	0.60	0.60	0.60	0.60
	Count, bag-of-words	0.49	0.60	0.53	0.56	0.45	0.50	0.52	0.52	0.52	0.52
	Count, POS	0.57	0.67	0.62	0.67	0.57	0.62	0.62	0.62	0.62	0.62
	Count, sentiment	0.52	0.72	0.60	0.64	0.43	0.51	0.58	0.57	0.56	0.56

	BOW, POS	0.60	0.66	0.63	0.68	0.61	0.64	0.64	0.64	0.63	0.63
	BOW, sentiment	0.54	0.64	0.59	0.63	0.52	0.57	0.58	0.58	0.58	0.58
	POS, sentiment	0.52	0.61	0.56	0.60	0.51	0.55	0.56	0.56	0.56	0.56
	Count, BOW, POS	0.58	0.67	0.62	0.67	0.58	0.62	0.62	0.62	0.62	0.62
	Count, BOW, sentiment	0.51	0.70	0.59	0.62	0.42	0.50	0.56	0.56	0.54	0.55
	Count, POS, sentiment	0.57	0.68	0.62	0.67	0.56	0.61	0.62	0.62	0.62	0.62
	BOW, POS, sentiment	0.52	0.62	0.57	0.61	0.51	0.55	0.57	0.57	0.56	0.56
	All features	0.57	0.67	0.61	0.66	0.56	0.60	0.61	0.61	0.61	0.61
Bi-LSTM	Baseline							0.66	0.93	0.77	0.71
	EMA							0.62	0.95	0.75	0.66
	FGM							0.70	0.81	0.75	0.71
	AUG							0.70	0.87	0.78	0.73
	EMA + AUG							0.72	0.80	0.76	0.73
	FGM + AUG							0.67	0.90	0.77	0.71
BERT	Baseline							0.82	0.92	0.87	0.86
	EMA							0.85	0.91	0.88	0.86
	FGM							0.83	0.92	0.87	0.86
	AUG							0.83	0.96	0.89	0.87
	EMA + AUG							0.84	0.94	0.89	0.87
	FGM + AUG							0.83	0.92	0.88	0.86
BERT of transfer learning	Baseline							0.72	0.88	0.79	0.76
	EMA							0.58	0.87	0.70	0.61
	FGM							0.69	0.85	0.76	0.72
	AUG							0.70	0.77	0.73	0.71
	EMA + AUG							0.66	0.85	0.74	0.69
	FGM + AUG							0.76	0.75	0.76	0.75

Table 6: Classification results

5.1 Summary of Results

The Table 7 above illustrates the performances of 6 basic classifiers based on 4 types of features and 11 feature combinations and 3 types of neural networks. We set linear regression based on word count feature as baseline, whose accuracy was 0.76. It can be seen in Table 8 that 16 models performed better or as well as the baseline. As expected, the best model that stands out was BERT using EMA + AUG method combination, with an overall accuracy of 0.87. Besides, the random forest model trained on word count, bag-of-words, part-of-speech, and sentiment features took the second place. For each of models except BERT of transfer learning, we used same training set and test set of fake news dataset.

For neural networks, except the best model - BERT using fake news dataset, the BERT of transfer learning, which was trained on cross-cultural US English dataset had a good performance, which is accuracy of 0.76, equal to the baseline. As to Bidirectional LSTM, its performance was quite normal, only attaining F1-score of 0.78 and accuracy of 0.73, below the baseline.

Order No.	Model	Feature/Augmentation Methods	Accuracy
1	BERT	EMA + AUG	0.87
2	Random Forest	All Features	0.78
3	Linear Regression	Bag-of-Words on TF-IDF	0.77
4	Linear Regression	Count, POS	0.77
5	Linear Regression	Count, Sentiment	0.77
6	Linear Regression	Count, BOW, POS	0.77

7	Linear Regression	Count, BOW, Sentiment	0.77
8	Random Forest	Count, Sentiment	0.77
9	Naïve Bayes	Count, Bag-of-words	0.76
10	Naïve Bayes	Count, BOW, POS	0.76
11	Linear Regression	Word Count	0.76
12	Linear Regression	Count, Bag-of-words	0.76
13	Random Forest	Word Count	0.76
14	Random Forest	Count, Bag-of-words	0.76
15	Random Forest	Count, BOW, Sentiment	0.76
16	BERT of Transfer Learning		0.76

Table 7: Top 16 models better than baseline

5.2 Evaluations

5.2.1 Evaluations for Neural Networks

From my perspective, BERT had a wonderful performance in our study has two main reasons, first is that it was pretrained on a large corpus of English data in a self-supervised manner. The corpus is BookCorpus, a dataset consisting of 11,038 unpublished books and English Wikipedia.

Consequently, it becomes very extensive and mature (Jacob et al. 2018). The second reason is that BERT has its core mechanism, attention mechanism. Generally speaking, attention mechanisms can help the model assign different weights to each part of the input, extract more critical and important information, and enable the model to make more accurate judgments without incurring greater costs for the calculation and storage of the model (Pengcheng et al. 2020).

According to Z. Cao et al. (2018), The lower performance (accuracy of 0.71) of Bi-LSTM is because LSTMs are affected by random weight initialization and so behave similarly to a feed-forward neural net. Instead, they prefer minimal weight initialization. Moreover, LSTMs are prone to overfitting, and the dropout approach is difficult to use to mitigate this problem. Dropout is a regularisation strategy that excludes input and recurrent connections to LSTM units from activation and weight updates when training a network.

For the experiment of model optimizations, it can be inferred from the result shown in Table 9 that different augmentation methods varied greatly, partly due to the relatively small amount of data itself, and partly because different models differed for different augmentation methods.

Note: Statistics are rounded to three decimal places.

	Baseline	EMA	FGM	AUG	EMA + AUG	FGM + AUG
LSTM	0.706	0.655	0.715	**0.732**	0.728	0.706
BERT + LSTM	0.863	0.864	0.855	**0.872**	**0.872**	0.86
Transfer Learning	**0.76**	0.61	0.72	0.71	0.69	**0.76**

Table 8: Results of neural networks' optimisation methods

- On the LSTM model, all the methods improved over baseline, except for the EMA method alone, which shows that the augmentation methods we designed worked. LSTM

model, using the data augmentation method brought the highest improvement, which was 0.26 points compared to baseline.

- On the BERT model, the augmentation methods did not improve except for the respective use of AUG and EMA. On the BERT model, the use of the data augmentation method and the EMA + AUG method resulted in the highest improvement of 0.09 points compared to the baseline. Compared with the LSTM model, it can also be seen that the BERT model was already strong enough, resulting in little improvement from our augmentation methods.
- On the transfer learning model, the various augmentation methods did not improve, but instead declined significantly. We suspect that this was due to the smaller amount of data.
- On the three models, it was found that the drop in test loss would be smoother with the model using EMA, suggesting that the generalisation ability of the model using EMA would be better.
- Overall, the direct use of data augmentation brought the highest improvement to the model, and was also the most convenient and quickest approach, so exploring good data augmentation methods afterwards is a research direction worth investigating. At the same time, we found that both EMA and FGM methods work best when used in conjunction with AUG, so EMA and FGM methods are also suitable for use when the amount of data is large. And using them when the amount of data is small will instead cause a decrease in results.

5.2.2 Evaluations for Basic Models

In terms of basic non-neural models, it can be easily seen from Table 8 that linear regression, random forest, and Naïve Bayes apparently appeared many times, which indicates those classification methods fitted the dataset well.

Random forest classifier constructed the features from three aspects: time feature, structural feature, and language feature, and trained a model based on these features to detect false

information (Hassan et al. 2017). More exactly, there are some advantages that contribute to the high performance:

1. It can handle very high-dimensional (a lot of features) data and does not need to do feature selection (because the feature subset is selected at random).
2. After training, it can give out which features are more important.
3. The training speed is fast, and it is easy to make a parallel method (the trees are independent of each other during training).
4. In the training process, the mutual influence between features can be detected.
5. For unbalanced datasets, it can balance errors.
6. If a large part of the features is missing, the accuracy can still be maintained.

About logistic regression, Jindal and Liu et al. (2018, 2010) used logistic regression models to use stylistic, metadata, and other features on the labelled Amazon reviews dataset, which is comparable to our fake news dataset, and finally combined with grammatical features to obtain AUC values of 63 % -78 %, proving that the logistic regression model's classification performance on this type of dataset is better than the support vector machine and the naive Bayes method. The result is similar to our experiment result. To explain it, Logistic regression is the preferred method for binary classification tasks. It outputs a discrete binary result between 0 and 1. It is very efficient, does not require much calculation, and is easy to understand, does not require scaling of input features, does not require any adjustment, and is easy to adjust, and outputs a calibrated prediction probability; another advantage of Logistic regression is that it is very easy to implement and very efficient to train (Bhattacharjee et al. 2017; Khurana and Intelligentie, 2017).

Another linear model Naïve Bayes also performed well, compared with other technologies, Naïve Bayes has its own strengths (Shu et al. 2018). For example, Yanmei et al. (2017) put forward the advantage of Bayes model in the research of network navy identification: when calculating the prior probability, the target values are independent of each other; based on the prior probability, the naval forces in the network can be calculated at the same time. The

classification of navy, which is not available in other methods such as decision trees and logistic regression; has good scalability, that is, when the extracted feature attribute set is optimized, the model is still valid. For another example, Narayanan et al. (2013) found that the use of Bayes model modelling, combined with negative processing, feature selection based on mutual information and other methods can effectively improve the accuracy of review classification, and the accuracy of the improved Bayes model can reach 88.80. %; Wu et al. (2008) improved the Bayes model to increase the accuracy of the new spam filtering technology by 8% compared to the traditional technology. In summary, it can be found that for information classification problems, the Bayes model has comparative advantages such as good adaptability, higher accuracy, and strong scalability.

Unexpectedly, also as a linear model, SVM model which is recognised suitable for text classification did not perform well (Cortes C.,1995), as delineated in Table 8, with an average of 0.65 among those 15 different features. The issue is probably that we only used the default parameters of the SVM without parameter optimization. A better approach is to construct the extracted features into feature vectors, use the grid search method to cross-validate to obtain the optimal parameters c and gamma, and replace the default c and gamma in the SVM classifier with the optimized parameters, then input the fine-tuned parameters into the SVM classifier for model training and get the final detection model (Yumei Fan et al. 2012).

To analyse features, it can be found that in most cases word count feature and BOW feature are better than POS feature and sentiment feature except KNN. Another observable finding is when combining word count and bag-of-words feature based on TF-IDF, the models tend to have a higher performance occasionally. This is an expected result as the two features complement each other (Van See. 2018).

Aside from that, it is also worth mentioning that for bag-of-words features, the lower n is, the better the performance would be. To explain it, when n is larger, there will be more binding information for the next word, greater discrimination, but sparser, and the total number of n-

grams is also more, which is V^n (V is the size of the vocabulary). Thereby, if the training dataset is very small, using a low-order n-gram model may do better. However, assuming you have enough data to avoid overfitting, then you can use higher-order models to get better data separability (William B. et al. 1994). The fake news dataset contains only 938 texts, so the lower-order gram is better.

6. Conclusions and Future Work

6.1 Conclusions

In conclusion, the study's aim was to apply AI algorithms to analyse deception in text and generalise across heterogeneous datasets, allowing for models that may be reused across multiple sources.

In order to attain the aim, a series of objectives has been completed, which comprised: collecting dataset, creating various features of the dataset, analysing the deception of the dataset, generalising across a different dataset and optimising the deep learning models.

Initially, the first goal was accomplished. Two datasets were gathered from different websites. More specifically, the fake news dataset includes real news texts as well as the counterparts of each real news text, namely, fake news texts. The dataset's real news was acquired from a variety of major news sources, especially in the United States, such as CNN, USAToday, NewYorkTimes, and so on. The false news in this dataset was produced by creating fake copies of the real news in the dataset using Mechanical Turk. As to celebrity dataset, the real news originated from major news websites. The fake news was obtained from gossip websites such as Entertainment Weekly, People Magazine, and other tabloid and entertainment-oriented publications. The news stories were gathered in pairs, with one being true and the other being fake. The pieces were cross-referenced with material from other online entertainment news sources and manually validated utilising gossip-checking websites like "GossipCop.com".

The second goal to create various features of the dataset was achieved as well. We generated 4 kinds of features, including bag-of-words based on word count, bag-of-words based on TF-IDF, syntactic POS feature, and sentiment lexicon feature. In terms of two types of bag-of-words, we applied word count vectorizer and TF-IDF vectorizer from scikit-learn library. As to syntactic POS feature, we obtained it by using a third-party tool named “ark-tweet-nlp”, a fast and robust Java-based tokenizer and part-of-speech tagger for texts, downloaded from http://www.cs.cmu.edu/~ark/TweetNLP/. Lastly, sentiment feature was implemented on the basis of the existing sentiment lexicon – AFINN. Finally, we integrated 4 individual features to 11 combinations in order to provide a better material for those classifiers of modelling step.

The third goal to analyse the deception was also completed. We trained 6 classic classification methods on 15 features or feature combinations by using scikit-learn, some of them performed quite well. Apart from that, we trained 2 deep learning models, which are LSTM and BERT separately.

The fourth goal to generalise what we had trained to the different dataset was realized as well. We stored the trained weights from the pre-trained BERT model, then used the weights on another BERT training the cross-cultural dataset, which led to a generic performance, which was the accuracy of 76%.

Last but not least, the final goal to optimise and augment the three deep neural networks was achieved. We introduced label smoothing and utilised multiple augmentation methods, encompassing EMA, FGM, AUG, EMA + AUG, and FGM + AUG, by which the performance and robustness of the models were improved to a certain extent.

It is worth mentioning that before using the effective strategies of label smoothing, EMA, FGM and AUG, I set several sets of training parameters and tried to add several layers of network structure hierarchy in order to optimise these deep learning models, and the performance of both LSTM and BERT did not improve after training. This is probably because the classical

network structure is already powerful enough and small changes to the network structure are of little significance, while innovation of the network structure is a more complex and time-consuming task for which we do not have enough time. Therefore, through various analyses and references (Yoo, J.Y., & Qi, Y. 2021), (Müller, Rafael, et al. 2021), (Chaudhary, Amit. 2020), (Miyato, Takeru, et al. 2021) and (Goodfellow, Ian J., et al. al. 2015), we have turned our thoughts to label smoothing, EMA, FGM and AUG as optimisation strategies with proven success.

Overall, the findings above of this study provide some insights of not only lie text analysis but also many of sentiment analyses of text. Firstly, we can generate syntactic POS feature and sentiment lexicon feature by using existing tools and data, which would like to strengthen our models. Secondly, Applying the previously trained model to other datasets through transfer learning can greatly reduce training costs and improve performance. Lastly, those optimisation strategies like data augmentation, EMA, and FGM can be used to enhance the models under different circumstances.

All in all, the results indicates that this project is a special and comprehensive investigation of lie detection in text, incorporating feature engineering, basic classification methods, deep learning classification methods, deep learning optimisation strategies, and transfer learning.

6.2 Future Work

6.2.1 Research Limitations

Nonetheless, though the project is generally solid and complete, it was limited by the absence of some aspects. Here are some limitations of research as follows.

Limitations	Recommendations
The punctuations, emojis and emoticons were not analysed because the dataset only contains words and basic punctuations.	If we collect a dataset containing abundant punctuations, emojis and emoticons about deception, such as tweets from Twitters, we would like to have filled the gap.

The fake news dataset incorporates fake news and celebrity news, whereas the cross-cultural English US dataset consists of three themes: abortion, best friend, and death penalty. But we did not analyse what themes would contribute to modelling.	We can take themes into account to analyse what differentiate among themes.
The project only focused on US English, didn't research on other English forms, like Indian English and Australian English, and other languages.	In the future, we can research on the dataset encompassing various languages or dialects, aiming to find the similarities and differences.
As for transfer learning, we only used our trained weights to generalise across one dataset.	We should generalise across diverse datasets, observing the performances.

Table 9: Research limitations

6.2.2 Technical Limitations

Due to the limited time available, there exist some technical deficiencies that if had been given more time, I would have improved. These technical limitations are shown in Table 10.

Limitations	Future Solutions
For dataset split, we utilized fixed training set and test set. I had tried K-fold and set n=5 on one classification method, but the performance was lower than fixed sets.	I would like to try different n values, and various parameter settings, to enhance the performance of models.
Some features had redundant dimensions. The uni-gram on TF-IDF had 2000 dimensions, and bi-gram on TF-IDF had 1500	In the future, I would like to select hundreds of best dimensions of those features with

dimensions. However, approximately 2/3 of the dimensions were extremely sparse, which would hardly contribution to the result.	redundant dimensions. In this way, the performance would probably be better.
The feature combinations didn't perform well. After discussion with schoolmates, the main reason may be the oversized and inconsistent scales.	We would like to apply standardization or normalisation, making sure the features of the feature combinations have identical dimension, thus have the correct weight contributions.
For SVM, we just used default parameter settings. Thereby, SVM didn't have expected ideal performance.	If time is enough, I am going to refer to Van Hee et al. (2018), using cross-validated grid search to get optimal parameters, which are 'C' and 'gamma'.
On the BERT of transfer learning model, the various augmentation strategies (EMA, FGM, AUG) did not improve but rather decreased, which is suspected to be due to the small amount of data (Cross-cultural English US Dataset - 458 pieces of text).	In the future I will gather small and large datasets of the same language and topic to do transfer learning experiments with these augmentation strategies for comparison and validation.
In that BERT is a complicated pre-trained neural network model, BERT expenses a lot. As a result, it could not be trained on normal PCs, so I turned to Colab, a free cloud platform provided by Google with high configured GPU, but it still cost 5 hours on training process.	In the future, if necessary, I would like to buy or rent a higher configured server for AI training.

Table 10: Technical limitations

7. Reflection on Learning

The moment I determined the project, I felt it would be a challenge for me, because prior to the project, I had seldom done projects of machine learning, not to mention neural networks. To accomplish it, I made a detailed plan under the advice of the supervisor.

At the beginning, I tried my best to supplement the theoretical knowledge of machine learning, neural networks, and NLP. Then, I went to get familiar with some common related tools and development libraries, such as: Jupyter Notebook, Pandas, scikit-learn, Keras, etc. Although these adequate preparations are time-consuming, they had laid a solid foundation for the progress of the later work.

After having basic research skills, I started to collect documents. First, my supervisor Prof. Yukun gave me a few common papers. I also found a lot of them from Google Scholar and Web of Science. After careful reading, I have a theoretical basis for lie detection in text. Next, I started to implement. However, I found that even with a thorough understanding of the theory, coding was not as simple as I originally thought, especially for neural networks because coding requires a lot of details, and it is easy to get bugs. After investigation, I chose the easy-to-use PyTorch framework which encapsulates many commonly used neural network functions, so it greatly improved development efficiency. In this way, I have realized the importance of tool selection. Proper tool selection can greatly improve research efficiency.

During the last month, Professor Yukun gave me a lot of inspiration in terms of technological innovation, enabling me to gain insights and incorporate my own innovative ideas in classical deep learning models, rather than just copying the original classical models. By constantly “searching for information - coming up with new ideas - implementing and validating”, the model performance was indeed improved, and both my creativity and technical skills have been improved considerably.

Time management and planning are very important. I didn't do enough about these. In the early stage of theoretical study, I spent a lot of time, and I had to take care of placement work,

PhD application, and moving. Therefore, in the practical stage, time seemed a bit rush. If I had made a detailed and adequate plan at the time and balanced the project with placement work and other affairs, then I could have done more of details about the aforementioned limitations and have reduced more minor deviations.

In addition, I gained a vital, transferable skill from this project: the "document-as-you-go" strategy. Many elements of this project, such as related work and descriptive analysis, were written as soon as they were processed. This not only made the compilation of this report easier at the end of the project, but the written parts were more accurate to the work had done at the time, as opposed to being written at a later stage when the logic for the work may have been forgotten.

In the end, I felt the challenge of self-supervision and motivation are one of the most essential things I gained during this project. The transferable abilities would be extremely beneficial in my further study.

To sum up, this project has been a success and I have been taught considerable meaningful lessons which would help me a lot in the future.

8. References

University Of Massachusetts at Amherst. 2002. UMass Researcher Finds Most People Lie in Everyday Conversation. ScienceDaily.

Levine, T. R., Serota, K. B., & Shulman, H. C. 2010. The impact of Lie to Me on viewers' actual ability to detect deception. Communication Research, 37, pp: 847–856.

Vosoughi, S. Roy D. Aral S. 2018. The spread of true and false news online. Science, 359(6380), pp: 1146-1151. doi: 10.1126/science.aap9559.

H. Zhang. 2004. The optimality of Naive Bayes. Proc. FLAIRS.

Rennie, J. D., Shih, L., Teevan, J., & Karger, D. R. 2003. Tackling the poor assumptions of naive bayes text classifiers. In ICML. Vol. 3, pp. 616-623.

C.D. Manning, P. Raghavan and H. Schütze. 2008. Introduction to Information Retrieval. Cambridge University Press, pp. 234-265.

A. McCallum and K. Nigam. 1998. A comparison of event models for Naive Bayes text classification. Proc. AAAI/ICML-98 Workshop on Learning for Text Categorization, pp. 41-48.

V. Metsis, I. Androutsopoulos and G. Paliouras (2006). Spam filtering with Naive Bayes – Which Naive Bayes? 3rd Conf. on Email and Anti-Spam (CEAS).

Rencher et al 2012. Linear Regression (Machine Learning). University of Pittsburgh.

David A. Freedman 2009. Statistical Models: Theory and Practice. Cambridge University Press. p. 26. A simple regression equation has on the right hand side an intercept and an explanatory variable with a slope coefficient. A multiple regression in right hand side, each with its own slope coefficient.

Rencher, Alvin C.; Christensen, William F. 2012. Chapter 10, Multivariate regression – Section 10.1, Introduction.Methods of Multivariate Analysis, Wiley Series in Probability and Statistics, 709 (3rd ed.), John Wiley & Sons, p. 19, ISBN 9781118391679.

Hilary L. Seal. 1967. The historical development of the Gauss linear model. Biometrika. 54 (1/2): 1–24. doi:10.1093/biomet/54.1-2.1. JSTOR 2333849.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. 1984. Classification and Regression Trees. Wadsworth, Belmont, CA.

J.R. Quinlan. 1993. C4. 5: programs for machine learning. Morgan Kaufmann.

T. Hastie, R. Tibshirani and J. Friedman. 2009. Elements of Statistical Learning, Springer.

Hastie, Trevor, Tibshirani, Robert; Friedman, Jerome (2008). The Elements of Statistical Learning (2nd ed.). Springer. ISBN 0-387-95284-5.

Piryonesi, Sayed Madeh (November 2019). Piryonesi, S. M. (2019). The Application of Data Analytics to Asset Management: Deterioration and Climate Change Adaptation in Ontario Roads (Doctoral dissertation) (Thesis).

Prinzie, A., Van den Poel, D. 2008. Random Forests for multiclass classification: Random MultiNomial Logit. Expert Systems with Applications. 34 (3), pp: 1721–1732. doi:10.1016/j.eswa. 2007.01.029.

Altman, N. S. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician. 46(3), pp: 175–185. doi:10.1080/00031305.1992.10475879.

D. Coomans; D.L. Massart. 1982. Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-Nearest neighbour classification by using alternative voting rules. Analytica Chimica Acta. 136: 15–27. doi:10.1016/S0003-2670 (01)95359-0.

B. L. Li, Q. Lu and S. W. Yu. 2004. An adaptive k-nearest neighbor text categorization strategy, ACM Transactions on Asian Language Information Processing. Volume 3, Issue 4, pp. 215–226.

Altman, N.S. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician. 46 (3): pp: 175–185. doi:10.1080/00031305.1992.10475879.

D. Coomans, D.L. Massart. 1982. Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-Nearest neighbour classification by using alternative voting rules. Analytica Chimica Acta. 136, pp: 5–27. doi:10.1016/S0003-2670 (01)95359-0.

B. L. Li, Q. Lu and S. W. Yu. 2004. An adaptive k-nearest neighbor text categorization strategy, ACM Transactions on Asian Language Information Processing, Volume 3, Issue 4, pp. 215–226.

Bottou, Léon, Bousquet, Olivier. 2012. The Tradeoffs of Large Scale Learning. In Sra, Suvrit; Nowozin, Sebastian; Wright, Stephen J. (eds.). Optimization for Machine Learning. Cambridge: MIT Press, pp. 351–368. ISBN 978-0-262-01646-9.

Bottou, Léon. 1998. Online Algorithms and Stochastic Approximations. Online Learning and Neural Networks. Cambridge University Press. ISBN 978-0-521-65263-6.

Spall, J. C. 2003. Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control. Hoboken, NJ: Wiley. pp. Sections 4.4, 6.6, and 7.5. ISBN 0-471-33052-3.

Robbins, Herbert; Siegmund, David O. 1971. A convergence theorem for non-negative almost supermartingales and some applications". In Rustagi, Jagdish S. (ed.). Optimizing Methods in Statistics. Academic Press. ISBN 0-12-604550-X.

Arjun Mukherjee and Vivek Venkataraman and Bing Liu and Natalie Glance. 2013. What yelp fake review filter might be doing. Proceedings of the International AAAI Conference on Web and Social Media. Washington, USA, pp:409-418.

Li J., Ott M., Cardie C., et al. 2014. Towards a General Rule for Identifying Deceptive Opinion Spam. Proceedings of Meeting of the Association for Computational Linguistics. Baltimore, USA, pp:1566-1576.

Depaulo B. M., Ansfield M. E., Bell K. L. 1996. Interpersonal Deception Theory. Communication Theory.6(3), pp:297–310.

Rayson P., Wilson A., Leech G. 2001. Grammatical word class variation within the British National Corpus Sampler. Language & Computers, Leiden, Netherlands:Editions Rodopi BV, pp:295-306.

Kaity M., Balakrishnan V. 2020. Sentiment lexicons and non-English languages: a survey. Knowl Inf Syst 62, pp: 4445–4480. doi: https://doi.org/10.1007/s10115-020-01497-6.

Ahire S. 2015. A survey of sentiment lexicons. Computer Science and Engineering IIT Bombay. Bombay.

Sun S, Luo C, Chen J. 2017. A review of natural language processing techniques for opinion mining systems. Inf Fusion 36, pp:10–25.

Giachanou A, Crestani F. 2016. Like it or not: a survey of twitter sentiment analysis methods. ACM Comput Surv (CSUR). 49(2), pp:28.

Medhat W., Hassan A., Korashy H. 2014. Sentiment analysis algorithms and applications: asurvey. Ain Shams Eng J. 5(4), pp:1093–1113.

Zhu X., Sobhani P., Guo H. 2015. Long short-term memory over recursive structures. International Conference on International Conference on Machine Learning, 2, pp: 1604-1612.

Pascanu R., Mikolov T., Bengio Y. 2013. On the difficulty of training Recurrent Neural Networks. International Conference on Machine Learning, pp: 2347-2355.

Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. 2017. Attention is All you Need. NIPS.

S. J. Pan and Q. Yang. 2009. A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering. vol. 22, no. 10, pp. 1345-1359. doi: 10.1109/TKDE.2009.191.

S. Hochreiter and J. Schmidhuber. 1996. Long Short-Term Memory. Neural Computation. vol. 9, no. 8, pp. 1735-1780. doi: 10.1162/neco.1997.9.8.1735.

Levine, T. R., Serota, K. B., & Shulman, H. C. 2010. The impact of Lie to Me on viewers' actual ability to detect deception. Communication Research, 37, pp: 847–856.

Bond, C. F. Jr., & DePaulo, B. M. 2006. Accuracy of deception judgments. Personality and Social Psychology Review, 10(3), pp: 214–234.

Bond, C. F. Jr., & DePaulo, B. M. 2008. Individual differences in judging deception: Accuracy and bias. Psychological Bulletin, 134(4), pp: 477–492.

Yoo, J.Y., & Qi, Y. (2021). Towards Improving Adversarial Training of NLP Models. ArXiv, abs/2109.00544.

Müller, Rafael, et al. When Does Label Smoothing Help? Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., 2021, pp. 4694–703.

Chaudhary, Amit. A Visual Survey of Data Augmentation in NLP. Amit Chaudhary, 17 May 2020, https://amitness.com/2020/05/data-augmentation-for-nlp/.

Miyato, Takeru, et al. Adversarial Training Methods for Semi-Supervised Text Classification. arXiv, 16 Nov. 2021. arXiv.org, http://arxiv.org/abs/1605.07725.

Goodfellow, Ian J., et al. Explaining and Harnessing Adversarial Examples. arXiv, 20 Mar. 2015. arXiv.org, http://arxiv.org/abs/1412.6572.

Fuller, C. M., Biros, D. P., & Delen, D. 2011. An investigation of data and text mining methods for real world deception detection. Expert Systems with Applications, 38, pp: 8392–8398.

Newman ML, Pennebaker JW, Berry DS, Richards JM. 2003. Lying Words: Predicting Deception from Linguistic Styles. Personality and Social Psychology Bulletin, 29(5), pp: 665-675.

Buller, D. B., Burgoon, J. K., Buslig, A., & Roiger, J. 1996. Testing Interpersonal Deception Theory: The language of interpersonal deception. Communication Theory, 6, pp: 268-289.

DePaulo, B. M., Lindsay, J. J., Malone, B. E., Muhlenbruck, L., Charlton, K., & Cooper, H. 2003. Cues to deception. Psychologi- cal Bulletin, 129, pp: 74-112.

Richards, J. M., & Gross, J. J. 1999. Composure at any cost? The cognitive consequences of emotion suppression. Personality and Social Psychology Bulletin, 25, pp: 1033-1044.

Duran, N. D., Hall, C., Mccarthy, P. M., & Mcnamara, D. S. 2010. The linguistic correlates of conversational deception: Comparing natural language processing technologies. Applied Psycholinguistics, 31, pp: 439–462.

Pennebaker, J. W., Francis, M. E., & Booth, R. J. 2001. Linguistic Inquiry and Word Count: LIWC 2001. Mahwah, NJ: Lawrence Erlbaum.

Ahmed, H., Traore, I., & Saad, S. 2018. Detecting opinion spams and fake news using text classification. Security and Privacy, 1(1), e9.

Mihalcea, R., & Strapparava, C. 2009, August. The lie detector: Explorations in the automatic recognition of deceptive language. In Proceedings of the ACL-IJCNLP 2009 conference short papers, pp. 309-312.

A. Arnold, R. Nallapati, and W. W. Cohen. 2007. A comparative study of methods for transductive transfer learning. Proceedings of the 7th IEEE International Conference on Data Mining Workshops. Washing- ton, DC, USA: IEEE Computer Society, pp. 77–82.

S. J. Pan and Q. Yang. 2010. A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359. doi: 10.1109/TKDE.2009.191.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In Proceedings of the 2017 conference on empirical methods in natural language processing, pp: 2832–2838.

Sebastian Schuster et al. 2018. Cross-lingual transfer learning for multilingual task oriented dialog. preprint, arXiv.

Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. 2017. Transfer learning for named-entity recognition with neural networks. preprint, arXiv.

Bryan McCann et al. 2017. Learned in translation: Contextualized word vectors. Advances in Neural Information Processing Systems.

Surafel M. Lakew et al. 2018. Transfer learning in multilingual neural machine translation with dynamic vocabulary. preprint, arXiv.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. preprint 1, arXiv.

Xiaodong Liu et al. 2019. Multi-task deep neural networks for natural language understanding. preprint, arXiv.

Colin Raffel et al. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. preprint, arXiv.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre and Rada Mihalcea. 2017. Automatic Detection of Fake News.

Verónica Pérez-Rosas and Rada Mihalcea. 2014. Cross-cultural Deception Detection. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 440–445, Baltimore, Maryland. Association for Computational Linguistics.

Van Hee, C., Lefever, E. & Hoste, V. 2018. Exploring the fine-grained analysis and automatic detection of irony on Twitter. Lang Resources & Evaluation 52, 707–731. doi.org/10.1007/s10579-018-9414-2.

K. Gimpel et al. 2010. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. Defense Technical Information Center, Fort Belvoir, VA. doi: 10.21236/ADA547371.

C. Van Hee, E. Lefever, and V. Hoste. 2018. SemEval-2018 Task 3: Irony Detection in English Tweets, in Proceedings of The 12th International Workshop on Semantic Evaluation, New Orleans, Louisiana, pp. 3950. doi: 10.18653/v1/S18-1005.

Kunneman, C. Liebrecht, M. van Mulken, and A. van den Bosch. 2015. Signaling sarcasm: From hy- perbole to hashtag, Information Processing & Management, vol. 51, no. 4, pp. 500-509. doi: 10.1016/j.ipm.2014.07.006.

F. Nielsen. 2011. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs, arXiv:1103.2903 [cs], Accessed: May 26, 2021. [Online]. Available: http://arxiv.org/abs/1103.2903.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Accessed: June 02, 2021. [Online]. Available: https://arxiv.org/abs/1810.04805.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. [Online]. Available: https://arxiv.org/abs/2006.03654.

Z. Cao et al. 2018. Improving Prediction Accuracy in LSTM Network Model for Aircraft Testing Flight Data. 2018 IEEE International Conference on Smart Cloud (SmartCloud), 2018, pp. 7-12. doi: 10.1109/SmartCloud.2018.00010.

Hassan, N., Arslan, F., Li, C., and Tremayne, M. 2017. Toward automated fact-checking: Detecting check- worthy factual claims by claimbuster. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1803–1812. ACM.

Jindal N, Liu B. 2008. Opinion spam and analysis. Proceedings of International Conference on Web Search and Data Mining. Stanford, USA, pp. 219--230.

Jindal N, Liu B, Lim E P. 2010. Finding unusual review patterns using unexpected rules. Proceedings of the ACM Conference on Information and Knowledge Management. Toronto, Canada, pp. 1549-1552.

Bhattacharjee, S. D., Talukder, A., and Balantrapu, B. V. 2017. Active learning based news veracity detection with feature weighting and deep-shallow fusion. In Big Data (Big Data), 2017 IEEE International Conference on, pp. 556–565. IEEE.

Khurana, U. and Intelligentie, B. O. K. 2017. The linguistic features of fake news headlines and statements.

Shu, K., Mahudeswaran, D., Wang, S., Lee, D., and Liu, H. 2018. Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media. [Online]. Available: https://arXiv preprint arXiv:1809.01286.

Yanmei Zhang, Yingying H., Shijie Gan et al. 2017. Research on Water Army Recognition Algorithm of Microblog Network Based on Bayesian Model. Journal of Communications. 38 (1), pp. 44-53.

Narayanan V., Arora I., Bhatia A. 2013. Fast and accurate sentiment classification using an enhanced naive bayes model. Lecture Notes in Computer Science, pp. 194-201.

WU J, DENG T. 2008. Research in anti-spam method based on Bayesian filtering. Proceedings of the 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application. Institute of Electrical and Electronics Engineers. Wuhan: IEEE Press, pp. 887-891.

Cortes C., Vapnik V.1995.Support-vector networks. Machine Learning. 20(3), pp. 273-297.

Yumei Fan, Chunjing Guo. 2010. Research and implementation of support vector machine algorithm. Journal of Hebei University of Engineering: Natural Science, pp. 106-112.

Cavnar, William B., and John M. Trenkle. 1994. N-gram-based text categorization. Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval. Vol. 161175.