ANN LAB: A Web application based on XAI concepts and implementing ANN models to solve NLP task

Yifan Pei

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science(Intelligent Systems)

Supervisor: Yvette Graham

September 2021

Acknowledgments

The first person I would like to thank is my mother, Mrs Lei Tang. She supported me in the completion of dissertation writing and, more importantly, without her financial help, I cannot even come to Trinity College Dublin and start my M.s.c program. The second person I want to thank is Professor Yvette Graham for her insight and guidance. In our biweekly discussion, we will talk about our progress so far and discuss the plan for the next step. Her professional and responsible attitude impressed me and motivated me to finish the dissertation. Then I would like to thank Professor Tim Fernando, my second reader. He gave me beneficial suggestions in the oral presentation, which helped me to increase the quality of my dissertation. Finally, I want to thank Mrs Kun Niu and Mr Mu Xu, who are my landlords. Due to the COVID-19 pandemic, their efforts to provide me with a quiet environment is touching.

Yifan Pei

University of Dublin, Trinity College
September 2021

ANN LAB: A Web application based on XAI concepts and implementing ANN models to solve NLP task

Yifan Pei, Master of Science in Computer Science

University of Dublin, Trinity College, 2021

Supervisor: Yvette Graham

Artificial Neural Network (ANN) is a kind of powerful Artificial Intelligence (AI) model inspired by the structure of a biological neural network and Explainable Artificial Intelligence (XAI) is a concept to emphasize the importance of the interpretability of the output of AI models for people. But, the potential of XAI in other fields, including education, can be still extracted since most applications of XAI are confined to researchers and developers, but the people who lack professional language cannot use applications based on XAI ideas to increase their knowledge.
This dissertation compares two existing projects: ANN Playground, developed by TensorFlow and Language Interpretability Tool (LIT) maintained by Google Research. We made a summary of their strengths and weaknesses and designed our Web application called ANN LAB. We choose a sentiment analysis problem, a subtype of Natural Language Processing (NLP), as the context of using four kinds of ANN models and showing the internal details. Besides, we have also implemented the visualization of the corpus called IMDB, which we use to train the models to allow users to gain knowledge about the impact the corpus will have on the performance of ANN models. Finally, to ensure the effectiveness of self-implemented ANN models, we choose AUC and ROC as the metrics to evaluate the models.

Summary

The primary purpose is to develop a Web application that applies the ideas from XAI to show the mechanism details of ANN models when using them to solve NLP or sentiment analysis problems.

We have developed a Web application that provides three services mainly: "corpus pre-analysis", "text visualization", and "ANN models". We make a simple introduction of the corpus that we use called IMDB. We randomly selected 5000-row data as the data set we will actually use.

We implemented three kinds of vectorization methods: TF-IDF, Doc2Vec and Word2Vec. While the factorized corpus is usually high-dimensional, in order to present the result of vectorization, we choose Uniform Manifold Approximation and Projection (UMAP) and Principle Component Analysis (PCA) to finish decomposition and put them in a 3D scatter chart. We also provide access for users to check the original review and processed review if they click the point in the visualized corpus in the scatter chart.

We used Keras to implement four ANN models: Multi-layer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Long Short-term Memory (LSTM).

To ensure the ANN models that the users can change the hyper-parameters including the selection of optimizer can solve the sentiment analysis successfully, we use receiver operating characteristic curve(ROC) and area under the curve (AUC) as the metrics to evaluate the ANN models, and the average AUC is 91.67%, and the worst one is

88.39% from RNN.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

According to the statistics of the social institution, the current market of Artificial Intelligence (AI) has developed rapidly. Based on the analysis and prediction finished by International Data Corporation (IDC)[12] (see Table 1.1). Meanwhile, the statistics portal Statista[12] expects that revenues from the AI market worldwide will grow fleetly. (Table 1.2).

Table 1.1: Summary and expectations from IDC about AI market

	2017	Expectation (2021)
Scale in US dollars	12 billion	52.2 billion

Table 1.2: Summary and expectations from Statista about revenue from AI industry

	2017	Expectation (2021)
Scale in US dollars	480 billion	2.59 trillion

Both statistics show that the scale of the current AI market and the expectation of future growth are optimistic. Although various AI algorithms appear powerful in results and predictions, they still suffer from low transparency, a Black-box problem. In other words, it is not easy to get insight into their internal working details.
Explainable Artificial Intelligence, also known as XAI, seized the interests of researchers, who are trying to increase the knowledge of ML algorithms working mechanisms. The Artificial Neural Networks (ANN) is a typical representation of the black-box problem since it is hard for researchers to determine the relationship between the input features and the output results.
Hence, we think the effectiveness of the combination of XAI and ANN would be an attractive point. We will be exploring the history of four kinds of ANN models and the cutting-edge research achievements in the fields of natural language processing (NLP) and sentimental analysis. Then we will introduce the design and implementation of our Web application: ANN LAB, which tries to provide an environment for users to learn the ANN models' details.
The main objective of this dissertation is by analysing the current research situation and two famous online tools based on ANN models and is driven by the ideas from XAI, make the summary about advantages and disadvantages, and start to state our research and Web application and how does our research can become a supplement for the current study.

1.2 Motivation

Although there are plenty of attempts and achievements in the field of XAI application, most of them are designed for researchers to get a better understanding of the

ANN they are working on and try to get better performance of a particular task, Lane et.al[13]lead research to come up with an XAI tool to give more effective after-action reviews (AAR) to make integration with an intelligent tutor into the XAI framework but did not focus on the interpretability of ANN model. Meanwhile, Suh et al. [14] tried to develop and justify the effectiveness of a risk calculator using XAI.
In conclusion, the current attempts in XAI are primarily for professional research and development for users who have no or limited knowledge of ANN models. the power that XAI has can be extracted furthermore.
On the contrary, for the XAI project, which does not require much-specialized knowledge, there are two projects which can be found on the Internet. One of them is the Language Interpretability Tool (LIT), a project maintained by Google Research.[15](see Figure 1.1)

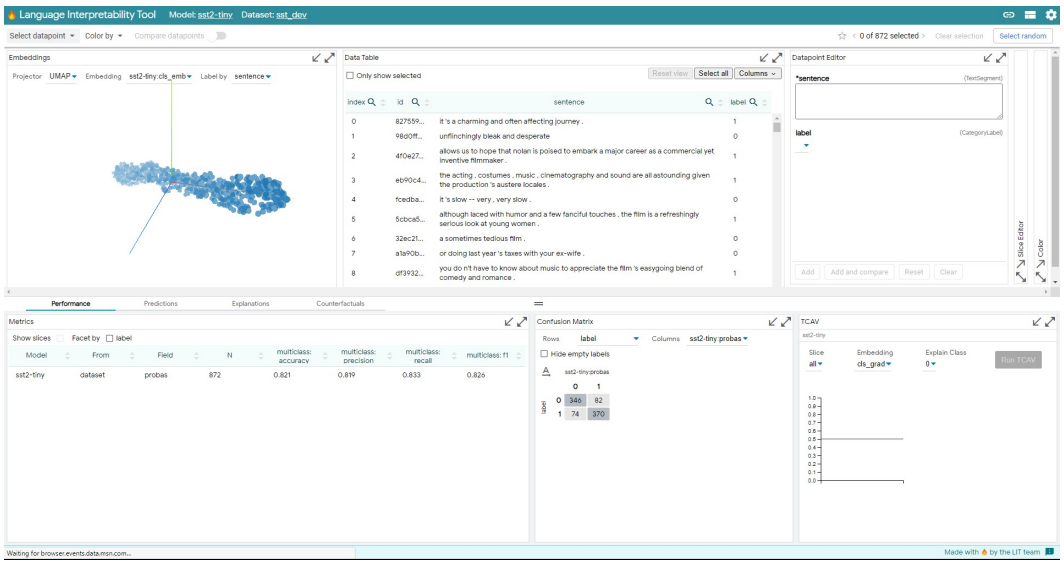


Figure 1.1: Demo of Language Interpretability Tool (LIT) hosted by Google Research

In Figure 1.1, it can be seen that Google Research has implemented an ANN model, which is not a simple re-implementation of the current classical ANN model, and the UI also shows the information such as each row data, the confusion metrics.
The other one is from the team of Tensorflow, which provides a more specific interface front-end page, and the progress of the training processing by the ANN model is more precise than the LIT. The name of their research is ANN Playground. (see Figure 1.2)

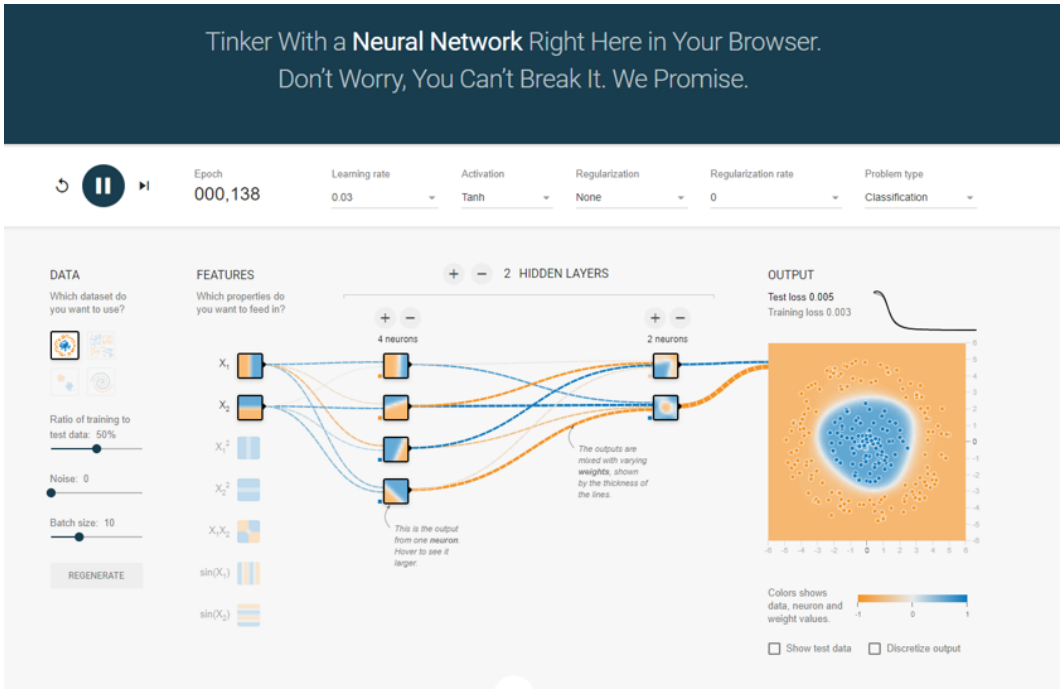


Figure 1.2: Demo of ANN playground designed by TensorFlow

In Figure 1.2, it can be seen that co the UI design of ANN Playground is more colourful and more impressively compared with LIT. ANN Playground provides the chance to manipulate the ANN model directly, e.g. Add or reducing the features from input layers and the number of hidden layers, and the visualization of the training details are much more explicit than LIT.
Based on these two kinds of research analysis above, the summary strengths and weaknesses of these two kinds of research can be seen below.

- LIT
  - Strengths
    1. Complete ANN model performance measurement indicators.
    2. Vectorization of the input data.
  - Weaknesses
    1. No explanation for the ANN model the project used.

2. Users cannot manipulate the ANN models but only watch the output from the same model.

- ANN Playground
  - Strengths
    1. ANN model training progress is visualized clearly.
    2. Adding or reducing hidden layers is available.
  - Weaknesses
    1. No visualization or explanation for the input data.
    2. Although the amount of hidden layers can be customized, the implementation of other ANN models (e.g. CNN) is not accessible.

1.3 Contributions

In this project, we implement four ANN models and designed an interactive user interface (UI), which allows the users to manipulate the pre-trained ANN models, including changing the hyper-parameters: optimizer, batch size and number of the epoch. Moreover, any user who wants to learn about ANN models can directly change our code after transforming our research into an open-source project.

1.4 Thesis Structure

In this thesis, the introduction of the research background will follow the first chapter, where we will state the situation of the current research background. Then the summary and comprehensive introduction of related academic publications, including essays and statistics report from the authoritative social institution. After the literature review, we will introduce the implementation details of our Web application, including the approaches to implement the ANN models. The introduction of experimental observation and evaluation will follow the last chapter. Finally, the last chapter will contain the summary of this dissertation, including limitations and possible future work.

Chapter 2

Literature Review

This section will introduce the previous research achievements and latest progress of the related academic field.

2.1 Natural Language Processing

The concept of NLP originally came from Information Retrieval (IR), which has the primary purpose of efficiently indexing and searching large amounts of text. More information and interpretation about IR can be found in the essay published by Manning et al. [16].

The early and simplistic application of NLP, which has been studied by Hutchins et al. [17], as an example, is verbatim and Russian-to-English machine translation. But, the appearance of disadvantages of dealing with homographs and metaphor has proved that relying solely on the specific meaning of the single word without any consideration about the relation between the words from context and the grammar details will lead to bad performance when there is a more casual or text with any kind of rhetorical device (e.g. metaphor).

After the attempt from Chomsky et al. in 1956 [18], which the authors explored multiple concepts of language structure and grammar, specifically English, and found several models that can generate new sentence from the original sentence which carries phrase structure based on formal properties of a collection of transformations of grammar. The conception of "formal properties of a collection of transformations of grammar"

has developed into "hand-written rules".
Because of the two features of natural language: size is extensive, unconstrained and ambiguous usage. The classical symbolic, hand-crafted rules for NLP task, on the one hand, it is possible that the rules themselves may lead to the unpredictable and unmanageable result of text meaning extraction. On the other hand, hand-write rules have poor performance when used in the casual and unconstrained language environment.
Based on these two questions, the technology of statistical NLP gradually matured, Lauer[19] designed a statistical processor stands for the NLP problem:

$$S = \langle \Omega, B, V, A \rangle$$

- Ω: set of all possible linguistic events

- $\langle B, V \rangle$: a finite set, the bins and values respectively

- A: the trained analysis function

With the development of AI and ANN, especially the implementation of "Embedding Layer" in ANN, which has the functionality of transforming the input text sequence of unequal length into a numerical fixed-length sequence[20], because of the features of the NLP task: ambiguous and variable inputs, natural language is symbolic and discrete, etc. Therefore, the ANN model is appealing to use in NLP.

2.2 Sentiment Analysis

For the definition of sentiment analysis, Poria et al.[21] mentioned that in 2016, which referred that sentiment analysis is the integrated application of NLP, or computational linguistics and text analysis.
Feldman[1] has made a summary about the applications of sentiment analysis, a structure of general sentiment analysis system is showed in their work as below:

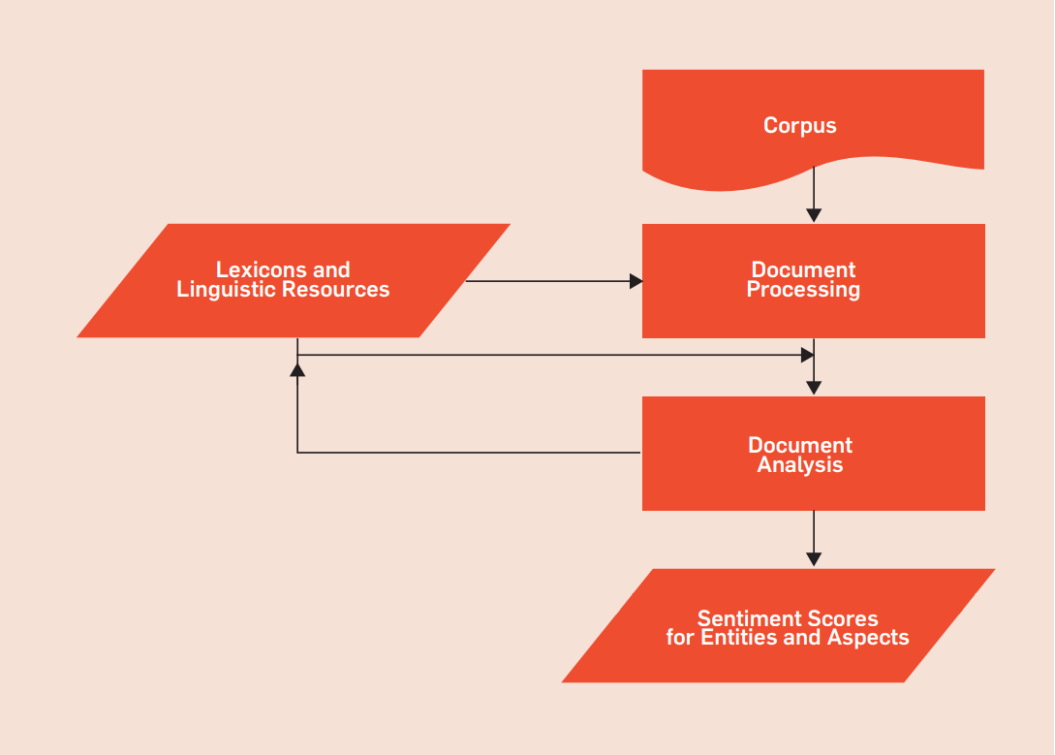


Figure 2.1: General sentiment analysis system structure[1]

There are three categories of questions level of sentiment analysis: document, sentence and aspect-based. The scale of the corpus decides the first two kinds, but when people are marking comments or reviews (see Figure 2.2), people usually tend to give their feedback of multiple attributes of the objective. Aspect-based methods are needed to solve this kind of sentiment analysis problem.

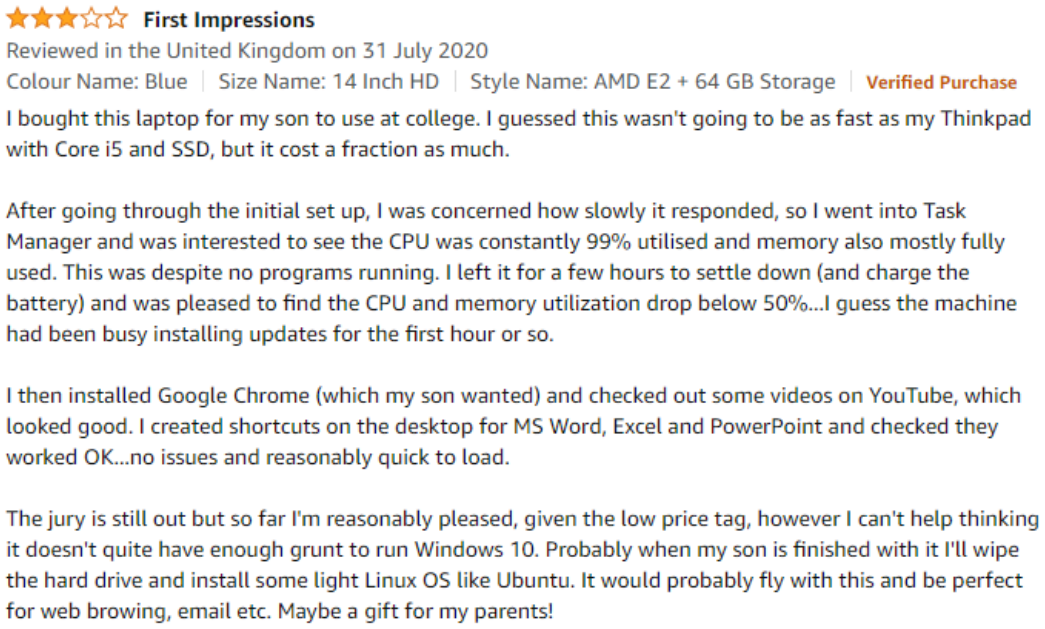


Figure 2.2: A Review in Amazon for a certain laptop

2.3 Artificial Neural Networks

ANN is firstly discussed in the pioneering work by McCulloch and Pitts[22] who has done massive mathematics reasoning. It can be seen that ANN is coming from the structure of brains as the name of it. The human's brain consists of billions of neurons responsible for receiving stimulation, generating excitement, and conducting excitement. More details about the human neurons are discussed by Hopefield [23]. He analyzes the feasibility of artificial neural signals from a purely mathematical and biological perspective and conducts experiments on this idea. More significantly, he comes up with the mathematical expression of a mathematical neuron as below:

$$y = \theta\left(\sum_{j=1}^{n} w_j x_j - u\right)$$

- n: Number of input data
- $x_j, j = 1, 2, ..., m$: input data

- $w_j, j = 1, 2, ..., m$: neuron's weight for the jth input
- u: threshold
- θ: unit step function at 0

Learning is a fundamental section of any intelligent system. Based on the equation above, Jain et al. argue two points to build a learning model: learning paradigm, which includes "supervised", "unsupervised", and "hybrid", and a learning algorithm[2]. Jain et al. also make a summary of the category of ANN models (see Figure 2.3)

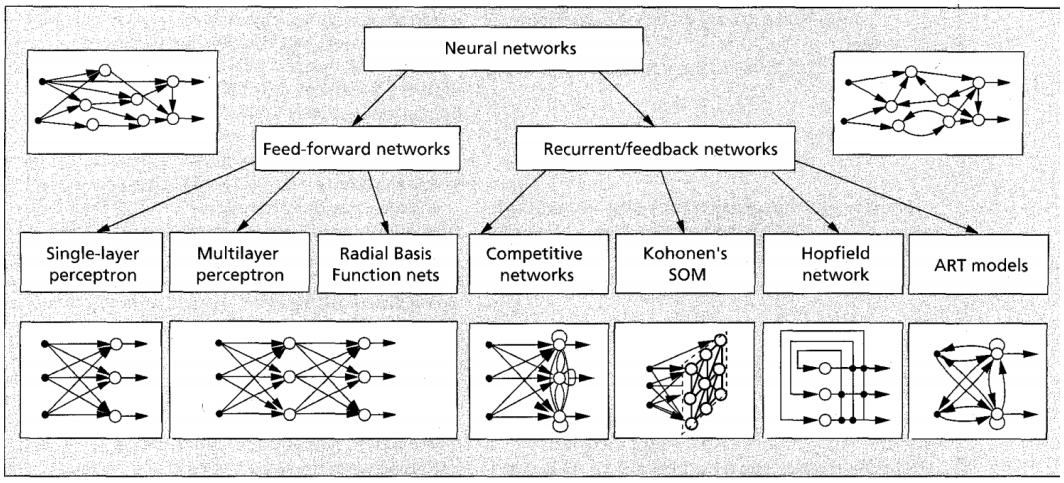


Figure 2.3: Category of ANN models[2]

There are two main sub-classes of ANN models, feed-forward and recurrent neural networks. In the feed-forward neural network, which can also be referred to as a forward neural network, while calculating the output value, the input value propagates from the input layer unit forward layer by layer, passes through the hidden layer and finally reaches the output layer, and obtains the output. The branches of the first layer of the forward network are connected to all the branches of the second layer, and the second layer is connected to the branches of the previous layer. There is no connection between the teams in the same layer. The excitation function of the neuron in the forward network can be represented by a linear hard threshold function or a non-linear function of unit rising, etc.

There are two important functions in the ANN model, the loss function and the activation function. The activation function acts on the data transfer process between

different layers of the neural network. If the activation function is not used in a multilayer neural network, each neural network layer is just a linear transformation. The multilayer input is also linearly transformed after being superimposed. Because the expressive power of the linear model is usually not enough, so the role of the activation function is reflected at this time, and the activation function can introduce non-linear factors. Using activation functions can guarantee to learn the sliding curve to divide the surface, instead of using complex linear combinations to force the sliding curve to divide the surface so that the neural network has a stronger representation ability and can better fit the target function. The common activation functions now include: $Sigmoid$, $ReLU$, $TanH$, etc.

The loss function is a basic concept in machine learning, statistics, probability and other research involving mathematical knowledge. From a mathematical point of view, the loss function maps the events of one or more variables to real numbers related to a certain cost. In supervised machine learning algorithms, we hope to minimize the error of each training example during the learning process. This is done using some optimization strategies such as gradient descent. Moreover, this error comes from the loss function. In other words, the loss function is used to evaluate the ANN models and the difference between the actual value y_i and the predicted value $\hat{y}$. Model performance increases as the value of the loss function decrease. If all possible output output vectors are $y_i = 0, 1$ and an event x with a set of input variables $x = (x_i, x_2...x_t)$, then the mapping of x to y_i is as follows:

$$\mathrm{L}\left(\hat{y}_i^{L+1}, y_i\right) = \frac{1}{t}\sum_{i=1}^{i=t}\left(y_i, (\sigma(x), w, b)\right)$$

Where $\mathrm{L}\left(\hat{y}_i, y_i\right)$ is the loss function. The common loss functions include: Mean Square Error (MSE), cross-entropy, etc.

Figure 2.3 shows that there are two main sub-classes of ANN. We also refer to the classification result of this picture. Furthermore, we implement two kinds of feed-forward networks and two kinds of recurrent networks. In the following subsections, we will introduce the history and state-of-art development of the four ANN models we will explore in this project.

2.3.1 Multiple Layer Perceptron

MLP is inspired by the Single Layer Perceptron (SLP) and perceptron learning algorithm. Even if SLP is regarded as a considerable advancement in the development of ANNN, the weaknesses of SLP still shows up with the development of ANN, in which one of the most obvious is that the ability to process non-linear data is weak[24]. Compared with the SLP structure, one of the most intuitive changes is that adding the hidden layer as a new middle layer. Thus, each neuron of the MLP can be regarded as an individual SLP to some extent, and the activation function is no longer limited to the Heaviside step function. The structure example can be seen in Figure 2.4.

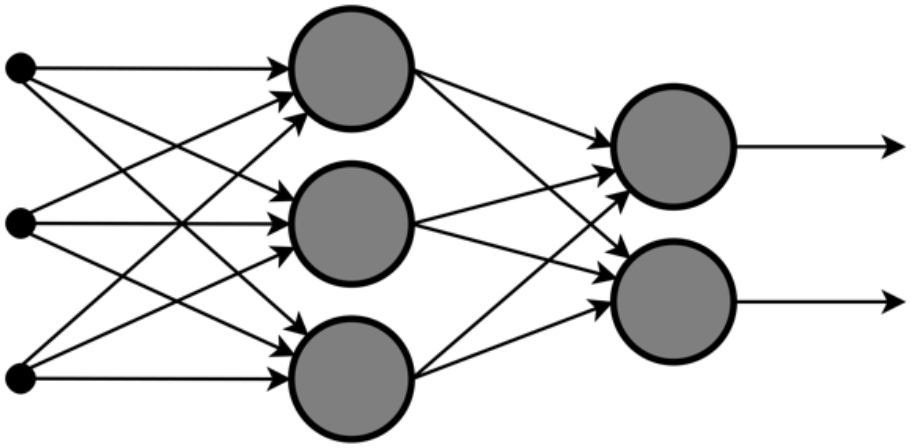


Figure 2.4: MLP model example

MLP as a neural network model developed in the 20th century, there have been many related applications, including applying the MLP model in drawing solar radiation maps in Spain by Hontoria, Aguilera, Zufiria[25] and Daniels and Kamp[26], where they build MLP networks to bound rating and house pricing.

2.3.2 Convolutional Neural Network

In the 1980s, Fukushima proposed "recognition" where he tries to build a network structure that can solve pattern recognition tasks, behaving like the human brain to help researchers understand the details of the human brain. He creatively introduces original ideas from the visual system of the human being to ANN, which can be con-

sidered the prototype of CNN. "Neocognitron" is an improvement of his previous work "Congnitron"[27], ten years later, as LeCun applied backpropagation to a network like "Neocoginitro" for supervised learning[28], the application of CNN has become more and more widespread, CNN has a good performance not only in the field of image recognition[29][30], but also in other fields like sentence modelling.[31]
Compared with MLP, CNN has two changes: the usage of convolution and pooling. In The operation of convolution between two functions is defined as:

$$(f * g)(\mathbf{x}) = \int f(\mathbf{z})g(\mathbf{x} - \mathbf{z})d\mathbf{z}$$

In other words, convolution measures the overlap between f and g. When we have discrete objects, the integral becomes a summation. For example: for the vector drawn from the set of infinite-dimensional vectors whose index is $\mathbb{Z}$, the sum of squares, we get the following definition:

$$(f * g)(i) = \sum_a f(a)g(i - a)$$

For a two-dimensional tensor, it is the corresponding sum of the (a, b), index of f, and $(i - a, j - b)$, the index of g :

$$(f * g)(i, j) = \sum_a \sum_b f(a, b)g(i - a, j - b)$$

The above is the description of the convolution operation in mathematics. In the convolutional layer, the input tensor and the kernel tensor are cross-correlation operations to produce the output tensor. So to some extent, the operation convolutional layer expresses a cross-correlation operation instead of convolution.
As for pooling operation, which is usually processed after convolution, the essence of pooling is sampling. Pooling selects a particular method to reduce the dimensionality of the input Feature Map to speed up the calculation. A more common pooling process is called Max Pooling. Assuming there is a 4*4 feature map used as the input feature for max pooling. A filter with a step of 2(the value of step can be chosen manually) will be used to scan the values in the neighbourhood of a feature map and select the maximum value to output to the next layer. (see Figure 2.5)

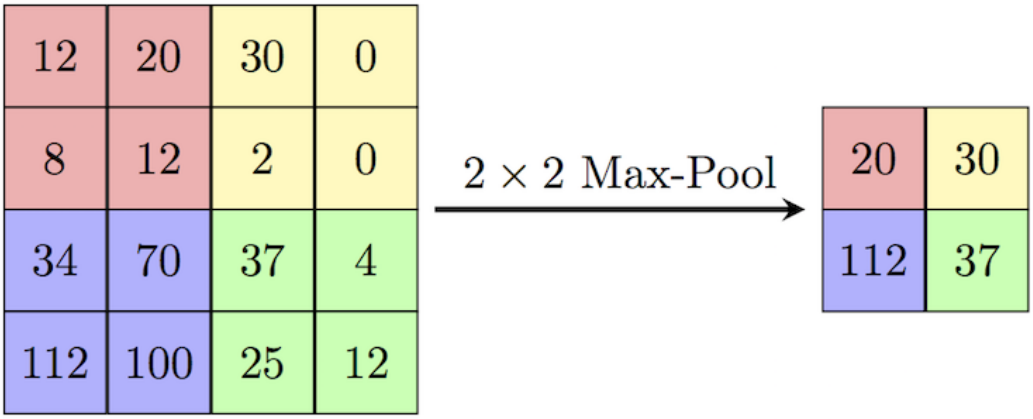


Figure 2.5: An example of max pooling operation

After all, the pooling layer is a process of feature selection and information filtering. In other words, the researchers have lost part of the information. This is a compromise with computing performance. With the continuous improvement of computing speed, this compromise is getting smaller and smaller.

2.3.3 Recurrent Neural Network

As we introduce the two ANN models before, all of them have a common feature: MLP and CNN are both feed-forward ANN models, one-to-one correspondence between input and output, that is, one input gets one output. There is no connection between different inputs, but when the target data is a sequence data – a series of interdependent data streams, using a feed-forward ANN model cannot satisfy this kind of data. For example, in predicting the next word in a sentence, it is not appropriate to take any of the preceding words. We not only need to know all the preceding words but also the order between the words. That is what problem the RNN wants to solve.
The earliest paper on the concept of RNN comes from Elman[32] in 1990. This paper mainly discusses how to find a specific pattern or structure in the time series. It is almost the original conceptual framework of RNN proposed in this paper. RNN was created to allow neural networks to have processing capabilities for sequential inputs. For example, to predict the next word of a sentence, it needs to be inferred based on

the previous words, and the order of the first few words is essential.
RNN is called "recurrent" because RNN performs the same operation on each sample in the input sequence, and the output of RNN is also based on previous calculations. Another way to think about RNN is that RNN is a learner with memory ability, which has a specific memory ability for the sequence calculated so far. The core content of RNN is to use recurrent links. After using recurrent links, the neural network can have dynamic memory capabilities.
The calculation process of RNN is as follows:

- x_t is the input vector of the input sequence at time t.
- s_t is the output of the hidden layer at time t, which is the memory unit of the RNN. The calculation of $s_t$is based on the output of the previous hidden layer s_{t-1} and the current input x_t.
- o_t is the output of the step moment t, for example, if you want to predict the next gait in the gait sequence, you can get: $o_t = \text{softmax}\,(Vs_t)$.

RNN is currently one of the most promising tools for deep learning. It solves the problem that traditional neural networks cannot share features that share locations from data. RNN has made many achievements in processing sequence data. Lin et al.[33] study the accessibility of applying RNN in document modelling. They summarise previous work on word coherence in different sentences and attempt to improve the quality of machine translation at the speech level. Then they design a hierarchical recurrent neural network language model for document modelling. An example of RNN is showed in Figure 2.6.

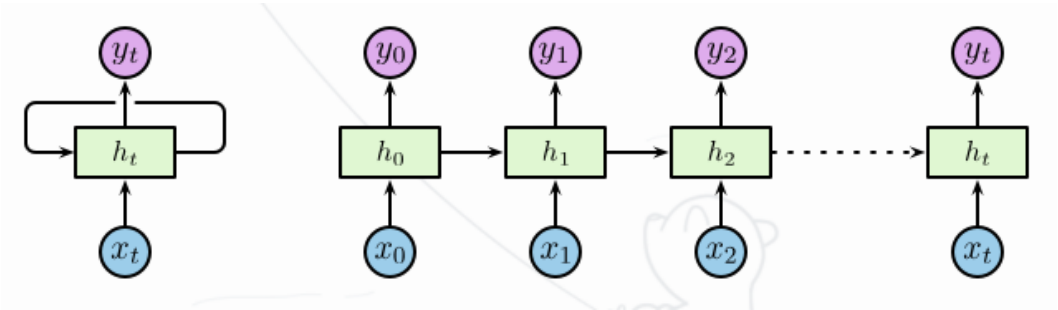


Figure 2.6: An example of RNN model

2.3.4 Long Short-Term Memory

Based on the introduction about RNN in the last section, the past output and the current input will be concatenated together in RNN. The output of the two is controlled through activation functions. RNN only considers the state at the most recent moment. In order to solve the disappearance of the temporal gradient in RNN, the machine learning field has developed a long and short-term memory unit, LSTM, which realizes the temporal memory function through the switch of the gate and prevents the disappearance of the gradient. We make a comparison between LSTM and RNN in Figure 2.7.

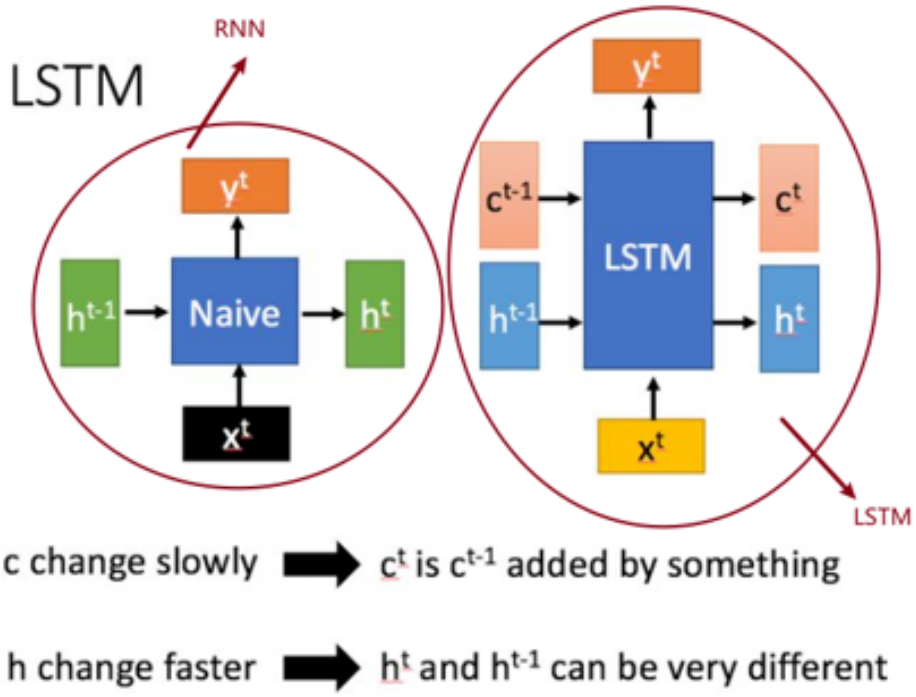


Figure 2.7: Comparison between RNN and LSTM

There are also study on the application of LSTM. Huang et al.[34] design a variety of traditional LSTM models solving the problem of document tagging. Specifically, besides the LSTM model, they also combine several LSTM-based models to get better performance. Another research result in the field of NLP with LSTM comes from Sundermeyer et al.[35]. Their paper summarises the problem that existed when using

RNN, and they use English and French corpus to analyze the LSTM model.

2.4 Explainable Artificial Intelligence

The term "Explainable Artificial Intelligence (XAI)" is firstly come up in the work of Van Lent et.al[36], they get the sponsorship from United States Department of the Army and state the AI architecture and possible explanation capability of an army training system called Full Spectrum Command (FSC) which is a training system used in US army at the moment. In their paper, The Explainable AI system in Full Spectrum Command is developed to be used in two phases: log the AI activities in the execution phase and use and analysis the logs in the after-action review phase.
Recently, there are no formal definitions of XAI recognized by researchers. There are two definitions mentioned in work finished by Emmert-Streib et.al[37], one of them is an indirect definition, which focuses on the functions and application of XAI in industry. The second one is a direct definition, point out the XAI is supposed to extract details and reasons to make itself apparent and easy to understand.
with the development of ML ANN in 1990s[38], the problem of "opacity", which is also called as black box problem, in AI algorithms has attracted the attention of researchers, but in fact at the very beginning, XAI serves the expert system and providing analysis of the reasons why the system makes decisions, in the paper from London[39], he point out the reasons why people do not only requires the accuracy of the model but also demand the model has the ability to justify its output and furthermore, he also describes the features of black box problem in deep learning, he argued that the researchers today can build the models very fast but it is difficult for users (or clients) to understand what happened inside the models and a more severe limitation is that besides the models cannot help clients understand the situation in problem, the features tracked by model are useless for users to understand the relationship between the actual features in the real world.
Using the ideas from XAI to build explainable machine learning models to increase the interpretability of AI models has become a field of extensive research to recognize the explainable machine learning models correctly. Fouladgar et al. [3] make a summary of the categories of explainable machine learning models. (See Figure 2.3)

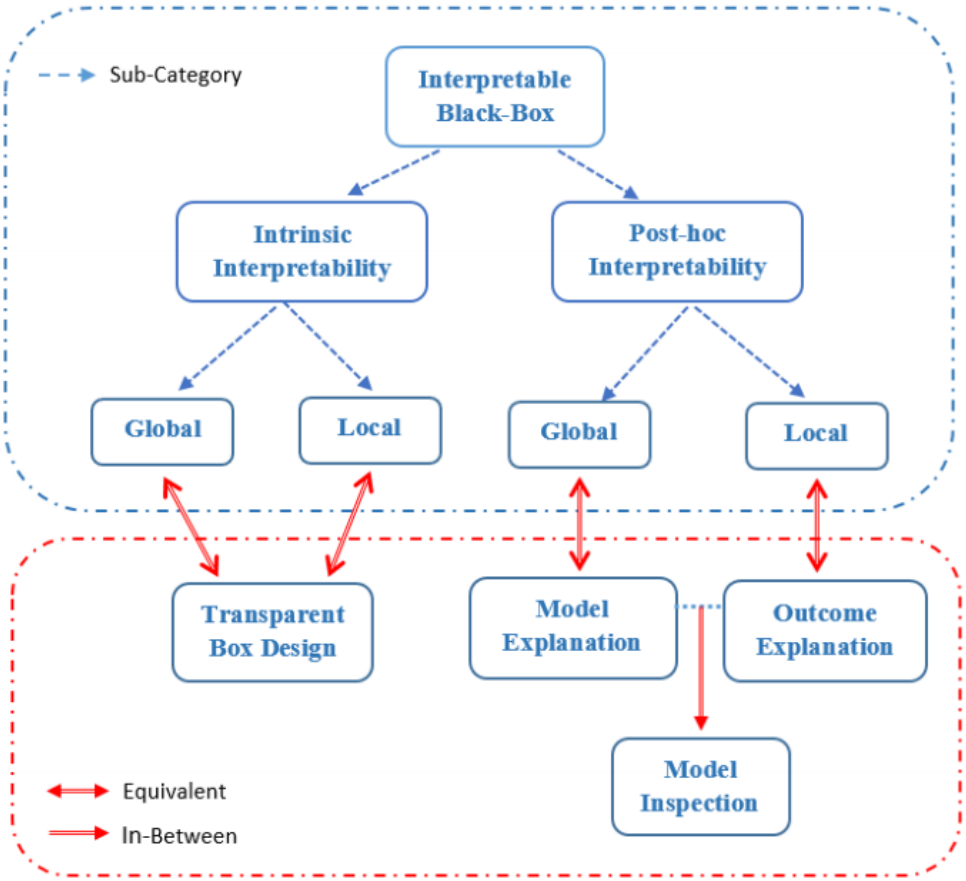


Figure 2.8: Categorization of explainable machine learning models[3]

The application of XAI already exists in many fields. Kuppa et.al[40] study the impact of applying XAI in the field of cyber security. In their work, they mention the term "Explainable Security (XSec)", which is created by Luka et al.[4]. Luka et al. draw a diagram that contains six "Ws" questions to describe the primary purpose and explain critical concepts in XSec. (See Figure 2.4)

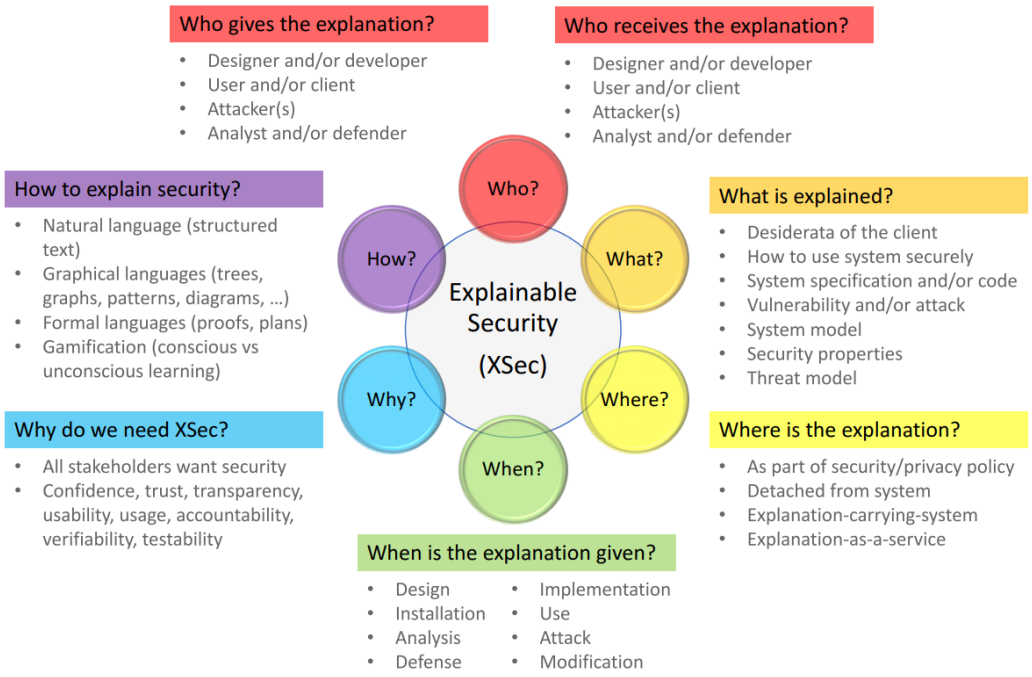


Figure 2.9: Six questions in XSec[4]

In Figure 2.4, it can be seen that all the aspects around XSec, including who are the potential consumers and how to solve the explainability problems. While in work from Kuppa et al., after they summarise the previous related work, they propose a method to classify XAI's dimension, which is related to the field of network security. They also design a novel black box in order to study security. Then another research processed by Deeks[41] proposes an idea that combing XAI and the judicial field. He agrees with the feasibility and necessity of introducing AI into the judicial process in previous related studies. However, he also points out that the judges should require reasonable explanation when they use the predictions or decisions made by AI models as the external aid to help them the sentence. Besides, he also mentions the usage of courts. For example, courts can be used to simulate and evaluate judicial AI algorithms. So, he believes that the exploration and study in applying AI in the judicial environment will also help the development and improvement of XAI.

Chapter 3

Implementation

In this chapter, we will introduce the implementation for each step of building ANN models and the front-end page.

3.1 Project Structure

After finishing the literature reading before starting researching, especially the summary of the strengths and weaknesses of the current two famous projects, we designed the overall structure of our project(see Figure 3.1)

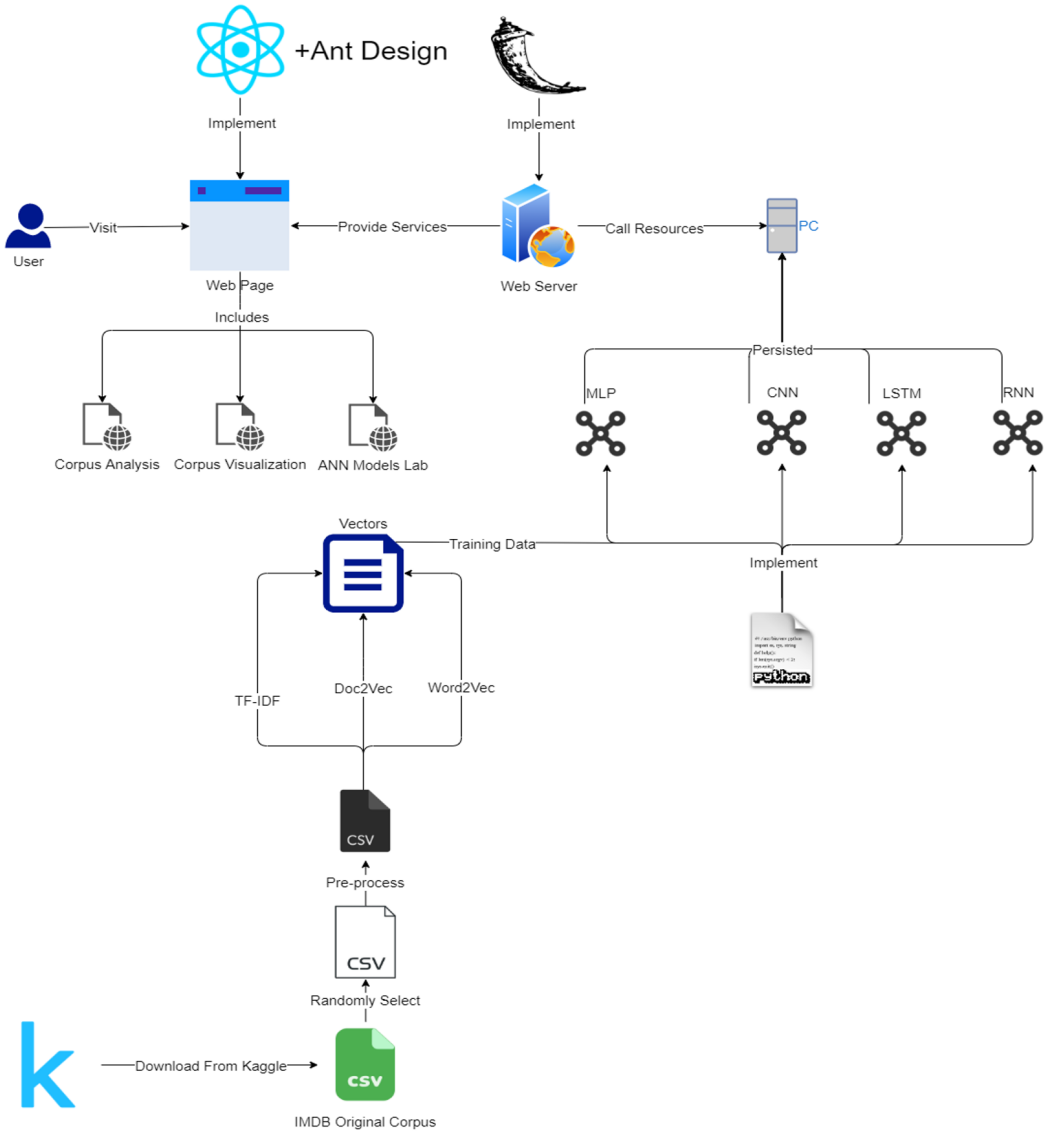


Figure 3.1: Project structure

In this project, when the user visits the front-end page, there will be a short introduction that refers to the main purpose of this project and information about the data set that is used in this project and the ANN models that we implemented.

3.2 Technical Implementation

In this section, we will introduce the technical implementation details in "corpus exploration and preprocessing", which contains the content of the basic introduction of the data set we will use and two brief descriptions of the data distribution characters of the data set. Then in the sub-section of "text visualization", we will focus on the methods that we use to create the vectors for the reviews data, reduce the high-dimensional text vector to 3 dimensions, and draw it on the scatter chart. In the third sub-section, "ANN Model Implementation", we will introduce the approaches to implement four ANN models and the corresponding overall structures. The following section will state how we build the route in the back-end server through Flask. Furthermore, in the last section, after the sections' introduction above, we will state the finalized project UI and introduce the framework that we use to build the front-end page at the same time.

3.2.1 Corpus Exploration and preprocessing

After obtained the IMDB corpus, basic observation can give us simple but useful information (see Table 3.1 and 3.2)

Table 3.1: Stat of Corpus Sentiment Polarity

	Positive	Negative
Amount	2517	2483

Table 3.2: Stat of Corpus Reviews Length

Review Length X	Amount
X <= 100	5
100 < X < 500	474
500 <= X < 1000	2101
1000 <= X < 1500	1014
1500 <= X < 2000	558
2000 <= X	848

It can be seen that in the corpus that we select, the distribution of sentiment polarity is balanced and avoid the problems caused by data imbalance. In Table 3.2, the most frequent comment length range is 500 <= X < 1000, and the second common is 1000 <= X < 1500, which can be evidence for the max vector dimension selection when we process the text vectorization mission.

After completing the evaluation of the corpus data, we need to process text preprocessing before vectorization.

The impact that text preprocessing has on the text classification has been studied before, in work from Uysal and Gunal[5], the made experiments on the Turkish and English corpus (see Figure 3.2)

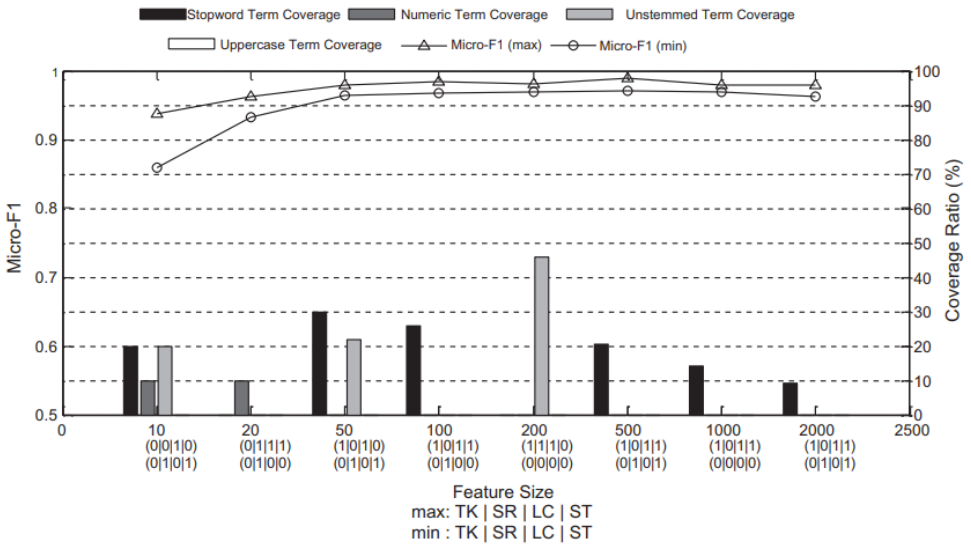


Figure 3.2: The experiments that text preprocessing has on English Emails corpus[5]

In Figure 3.2, the graph has proved that the performance (Micro-F1) of text classification without any text preprocessing is the worst. The performance gets better when text preprocessing is applied. However, it can be seen that "Numerical Term Coverage" becomes 0% after the Micro-F1 reaches the maximum, and there is no noticeable performance changing whatever the coverage of unstemmed and stopword terms change.

Vijayarani and Nithya et.al[6] has come up with a flow chart with text preprocessing (see Figure 3.3)

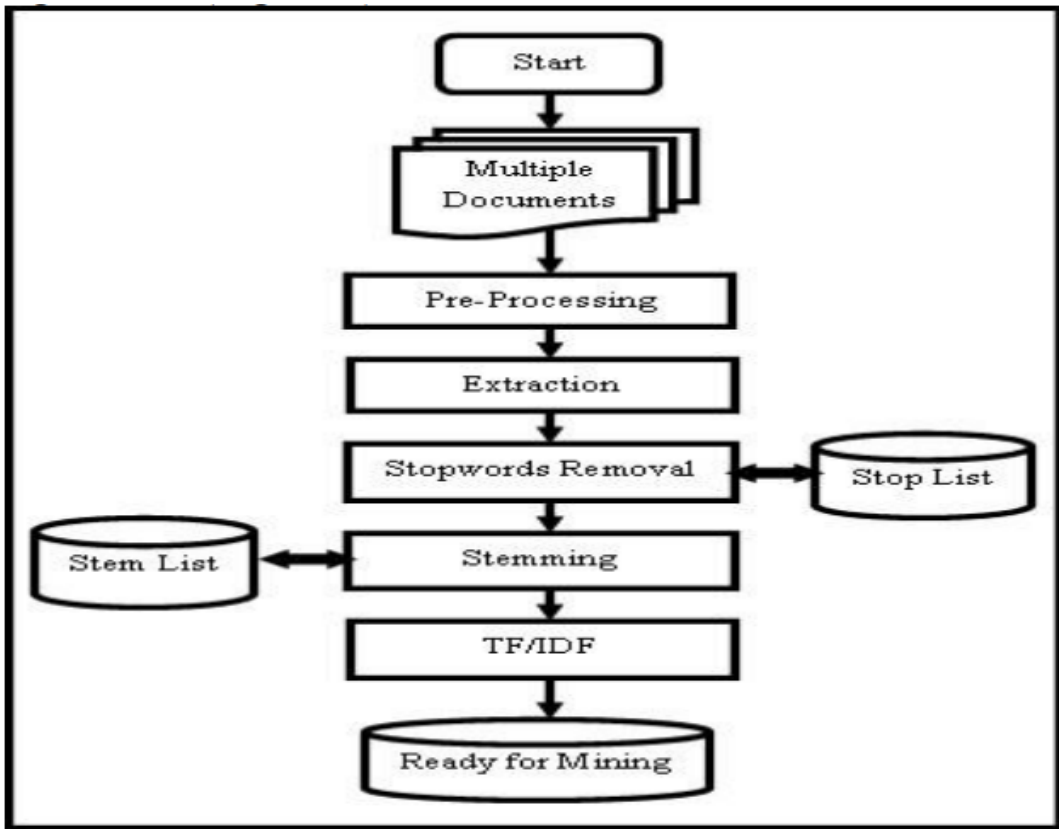


Figure 3.3: Text mining preprocessing techniques[6]

Generally, text preprocessing contains a series of recognized operations[42], including tokenization, stemming, lemmatization and removing stop words, etc.

Tokenization

The purpose of tokenization is to transform the original string text into a list contains separated elements. In our case, since the corpus we use is in English, tokenization becomes easy since English text is already separated by spaces, calling the *split*() function is a fast implementation for tokenization in our project.

Stemming

The purpose of stemming is to remove the prefix and suffix of a word to get the root. The first stemming algorithm is come up by Lovins[43] in 1968, which attempts to elim-

inate common suffix including -ses, -ing or -ation. The derived stemming algorithms like Porter[44], which has multiple advantages compared with Lovins's method, e.g. an extraordinary complexity reduction in suffix removal. Porter stemming algorithm has been proved as a effective stemming method in real application[45][46][47][48].
In our project, we directly use the function of *PorterStemmer*() from *nltk* library to implement stemming.

Lemmatization

Lemmatization has standard features with stemming[49], the purpose of both of them is to find the original form of a word and try to avoid the deviations in text analysis and semantic understanding caused by different forms of words, but compared with stemming, lemmatization can ensure that the outcome is still an accurate, meaningful word while stemming algorithm may cause inexplainable result, an example mentions a possible situation in Figure 3.4.

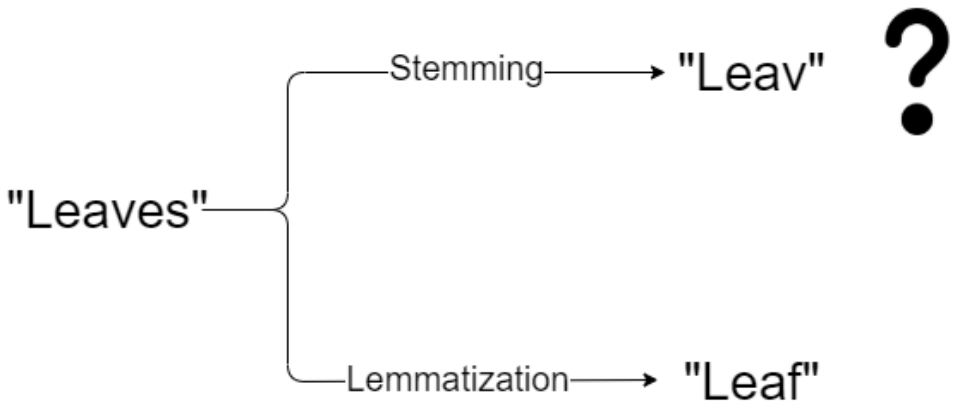


Figure 3.4: A possible situation when applying stemming and lemmatization

The positive influence of using lemmatization in NLP has been supported in many applications of NLP. Korenius and Laurikkala[50] made a series of experiments with the lemmatization method in Finnish text documents, and the performance is good in their 5000 experimental documents.

One of the reasons the lemmatization can guarantee the output is the meaningful word is that it uses a certain database as the reference. In our project, we use $WordNetLemmatizer()$ function from *nltk* library to finish the lemmatization task. WordNet is the database that the lemmatization algorithm will use. It is a lexical database that collects words and clusters four forms of a lexicalized concept: noun, verb, adjectives and adverbs into sets of synonyms[51]. $WordNetLemmatizer()$ function is an existing implementation of applying WordNet database in lemmatization.

Remove Stop Words

In our sentiment polarity problem, thea basic idea is to evaluate and calculate the single review data into a high-dimension vector and send these vectors into ANN models, but as a matter of fact, not every word in a sentence has an influence on the sentiment of the sentence and these words can be removed. These words can be called as stop words, e.g. "the", "if", "but", "and"[52]. The concept of "stop word" does not only contain the actual word, but punctuation marks in various languages are also often used as part of stop words like ",", ".".
The impact of removal based on the stop words list has been proved in actual researches. El-Khair[7] uses standard recall and precision to measures the performance of using different stoplists in Arabic information retrieval missions. (see Figure 3.6)

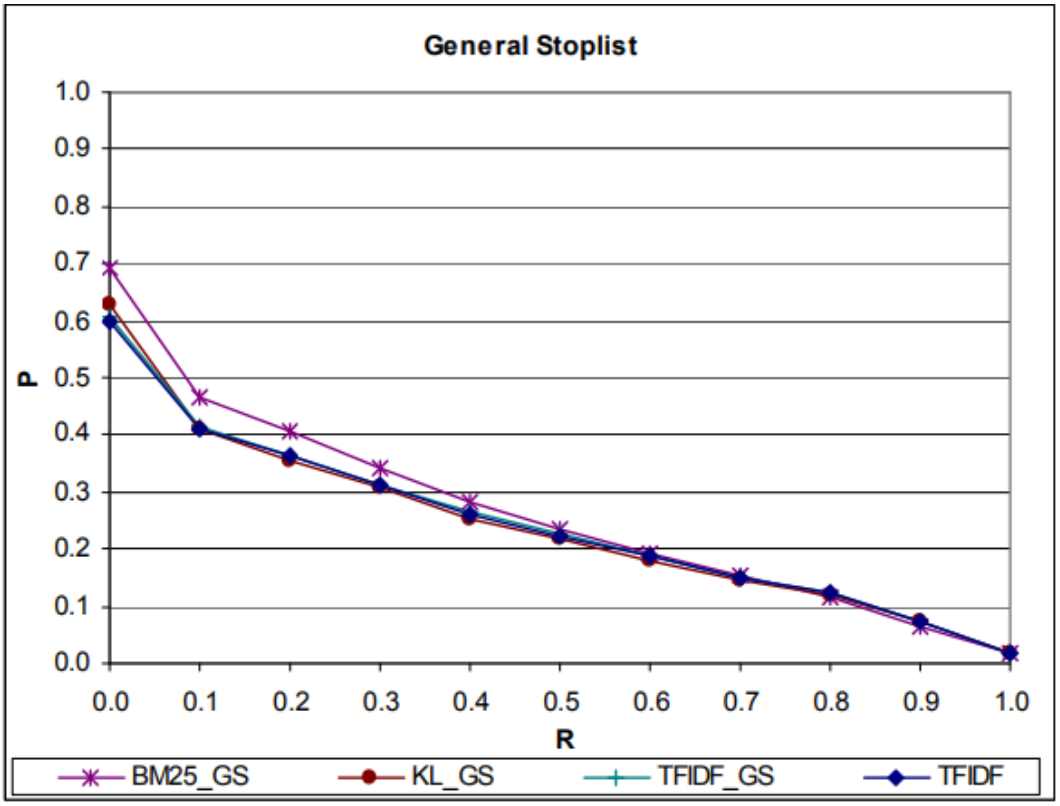


Figure 3.5: Recall-Precision curves for the general stoplist[7]

Because the people's reviews in our data set do not belong to a particular field, building a targeted stop words list is unnecessary in our case. We use a pre-compiled stop words corpus from library *nltk* to eliminate the stop words in the review data from the data set.

In our project, since the text is crawled directly from the Internet, which causes that the reviews data contains HTML tags, e.g.
. As the solution, besides the joint operations in text preprocessing, the clean HTML tags are included in the implementation of our text preprocessing. Besides, some reviews contain emoticons made up of characters, e.g. ":)". These strings are not actual English words, but we also believe they are not meaningless characters that should be removed like stop words. For example, ":)" represents that the corresponding review is more likely to be positive. To solve the problem of processing emoticons, in our text processing function, the measures we

have taken are basically to retain these meaningful but not specific strings of English words to keep the emotional meaning in these unique strings as much as possible. We use the regular expression from the *re* library from Python to find the HTML tags in the text and remove them in our custom regular expression rules. The specific code details related to processing special characters, please refer to appendix **??**.
The sequence diagram of text preprocessing is shown in Figure 3.7. The overall text processing function code details of preprocessing can be found in appendix **??**.

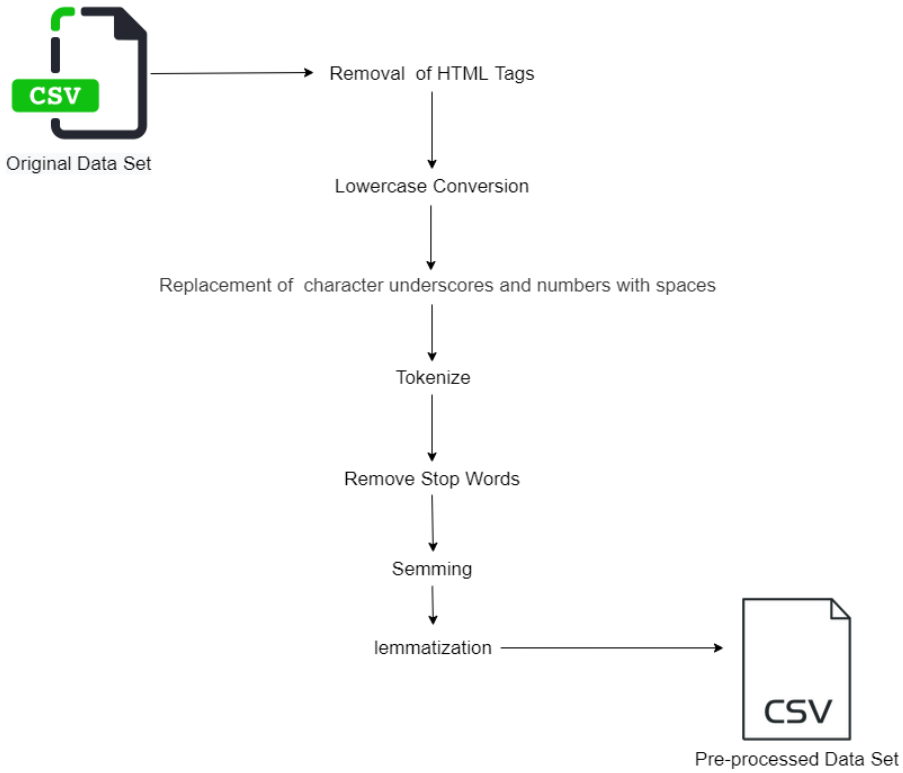


Figure 3.6: Sequence diagram of text preprocessing

3.2.2 Text Visualization

In this section, there are two sub-sections: the first is the introduction for the methods that we use to implement three kinds of vectorization, then the description of

decomposition is included in the second sub-section.

Vectorization

In our project, to extract more information beneath the corpus, we used three kinds of methods of vectorization: TF-IDF, Word2Vec and Doc2Vec.
TF-IDF is a weighting technique used for information retrieval and data mining. There are two keywords: "TF" and "IDF" TF (Term Frequency) means to count the number of times a word appears in the target document. However, the weakness of using TF to describe the relevance of documents relativity ultimately is found soon. Because of language factors, some words do not have actual meaning but play the role of connection in the sentence. Besides, When the source of the text type is more complex, such as speech text or some over-spoken text, the effect of using TF alone on this type of corpus is very bad. At this time, TF cannot help us distinguish the relevance of documents. The concept of IDF (Inverse Document Frequency) was put forward in response to this problem. IDF method points out the necessity of penalizing words that appear in too many documents. If a word is more common, the denominator is more significant, and the inverse document frequency is smaller and closer to 0. The reason for adding 1 to the denominator is to prevent the denominator from being 0 (that is, all documents do not contain the word). TF-IDF method is firstly come up from the research from Jones in 1972[53] and 2004[54]. The classical formula used in TF-IDF to calculate the term weight is as below:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

- $w_{i,j}$: the weight of term i in document j.
- $tf_{i,j}$: the frequency of term i in document j
- N: number of documents in the data set.
- df_i: number of documents containing term i

The advantage of TF-IDF is that it is simple, fast, and easy to understand. The disadvantage is that sometimes the word frequency is used to measure the importance of a word in an article is not comprehensive enough, and sometimes essential words may not appear enough. This calculation cannot reflect the location information and the

importance of the word in the context. If the context structure of the word needs to be reflected, then other algorithms like Word2Vec can be an option. Word2Vec is a tool that provides a new approach compared with TF-IDF called the word embedding model for transforming the word into vector space. It is introduced by Mikolov et al. [55] from Google originally. By training the Word2Vec model, the researchers can regard the weight matrix from the hidden layer as the vector for the specific word. There are two methods to get the matrix (vector): continuous bag-of-words, referred to as CBOW, and continuous skip-gram. Continuous skip-gram is a method to predict the word given based on the context and surrounding words. There is an example task for continuous skip-gram. (see Figure 3.2)

Source Text	Training Samples generated from source text			
I will have orange juice and eggs for breakfast	(will, I)	(will, have)	(will, orange)	
I will have orange juice and eggs for breakfast	(have, I)	(have, will)	(have, orange)	(have, juice)
I will have orange juice and eggs for breakfast	(orange, will)	(orange, have)	(orange, juice)	(orange, and)
I will have orange juice and eggs for breakfast	(juice, have)	(juice, orange)	(juice, and)	(juice, eggs)
I will have orange juice and eggs for breakfast	(and, orange)	(and, juice)	(and, eggs)	(and, for)
I will have orange juice and eggs for breakfast	(eggs, juice)	(eggs, and)	(eggs, for)	(eggs, breakfast)
I will have orange juice and eggs for breakfast	(for, and)	(for, eggs)	(for, breakfast)	

Figure 3.7: An example of the task for continuous skip-gram

On the contrary, CBOW is a method to use the predict current value through context. The comparison is showed in Figure 3.3

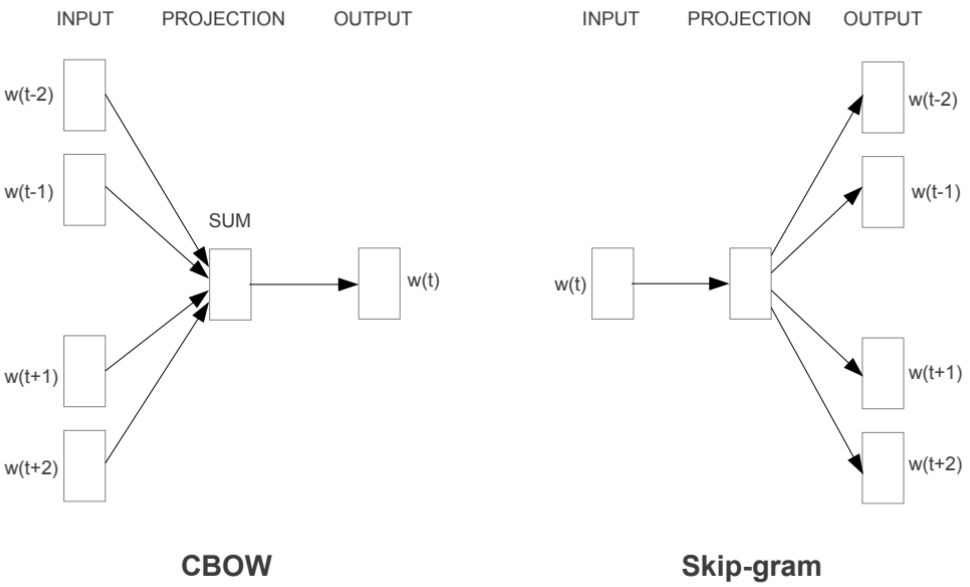


Figure 3.8: Structure diagrams of CBOW and continuous skip-gram [8]

The last vectorisation tool we used is called Doc2Vec, which is proposed by Le and Mikolov [56]. Word2Vec can be regarded as an extension for Word2Vec n the aspect of document-level corpus since Word2Vec attempts to preserve more information during vectorisation by implementing simultaneous learning of both words and documents. [57]

In our project, text vectorisation aims to present the low-dimensional visible relation between the reviews from the data set. Considering the hardware we are using, too many features will cause the memory to exceed, but the small number of features will cause losing features during vectorisation. Therefore, we used 1000 as the size of the vector to maximise the information we can retain. Then we store the vectorisation result as a .npy file to prepare for the presentation in the front-end UI. We choose the implementation of the TF-IDF vectorisation method as an example in appendix **??**.

Decomposition

The output of vectorization is a one-to-one high-dimensional vector corresponding to the original review data. Naturally, we cannot use these vectors to finish visualization

directly. We consider using decomposition methods to map high-dimensional vectors to low-dimensional 3D space and draw the scatter graph with a front-end framework. The concept of decomposition is coming from mathematics and statistics. In the field of AI, the purpose of decomposition is to learn a mapping function $f : x- > y$, where x is the original high-dimensional vectors and y is the low-dimensional vector after function f processes.

We select two kinds of decomposition methods: Principle Component Analysis (PCA) and Uniform Manifold Approximation and Projection (UMAP).

The essence of principal component analysis is to treat the data as an ellipsoid. In two dimensions, it is an ellipse. We know that the ellipsoid has a major axis and a minor axis. If a short axis of a 3-dimensional ellipsoid is very short, can we regard this short axis as not? Such an ellipsoid is reduced to a 2-dimensional ellipse. This is the principle of PCA. To facilitate calculations. First, you need to normalize the data so that the centre of the ellipse is at the origin**. So how do we know which minor axis is very short? Look at the variance of this axis direction data. The smaller the variance, the shorter the axis. That is to say, the larger the variance, the more important the data of this dimension is the principal component. The smaller the variance, the less important this dimension (corresponding to the shorter the ellipse axis), then this dimension can be discarded. The data with the largest variance is projected to the first coordinate (also called the first principal component), the second-largest is projected to the coordinate of the second dimension, and so on. More information can be found in the works from Ringnér[58] and Karamizadeh et.al[59].

The PCA algorithm has been applied in many AI applications. For example, Destefanis et.al[9] uses the PCA method in the mission of beef characterization. They define multiple PCA tools and compare them to find the most influential characteristics based on eigenvalues (see Figure 3.9).

Component	Eigenvalues	% of variance	Cumulative variance,%
1	6.10	33.90	33.90
2	3.72	20.64	54.54
3	1.43	7.94	62.48
4	1.36	7.54	70.02
5	1.20	6.69	76.71
6	1.00	5.58	82.29

Figure 3.9: Results from the principal component analysis for the first six principal components[9]

An early technology called t-SNE is also a kind of non-linear dimensionality reduction technology which is brought by Van der Maaten and Hinton [60]. t-SNE is an upgraded version of SNE. It maps data points to probability distributions through affinity transformation. Although SNE provides a suitable visualization method, it is challenging to optimize, and there is a "crowding problem" (crowding problem). T-SNE uses a heavier long-tailed distribution of t distribution in low-dimensional space to avoid crowding and solve optimization problems.
UMAP is an attempt to overcome those weaknesses, which is a state-of-art non-linear dimensionality reduction technology directly coming from the work of McInnes et.al[10]. In their essay, UMAP is constructed based on the theoretical framework of Riemannian geometry and algebraic topology. UMAP has apparent advantages, faster running speed, and a small memory footprint when processing large data sets. UMAP is a dimensionality reduction technology, similar to t-SNE. This algorithm is based on three assumptions about the data:

- The data is evenly distributed on the Riemannian manifold (Riemannian manifold)
- The Riemann metric is locally constant (or can be approximated this way)
- Manifolds are locally connected

The name of the first assumption is related to a mathematical concept, "manifold", which refers to areas connected. Mathematically, it refers to a group of points, and each point has its neighbourhood. Given any point, its manifold locally looks like Euclidean space. In other words, it has the properties of Euclidean space in its local space and can use Euclidean space for distance calculation. Therefore, it is easy to establish a dimensionality reduction mapping relationship locally and then try to generalize the local relationship to the global, and then visualize it. The pseudo-code of the UMAP algorithm is shown in Figure 3.10.

Algorithm 1 UMAP algorithm

function UMAP(X, n, d, min-dist, n-epochs)

 # Construct the relevant weighted graph
for all $x \in X$ **do**
    fs-set$[x] \leftarrow$ LocalFuzzySimplicialSet(X, x, n)
top-rep $\leftarrow \bigcup_{x \in X}$ fs-set$[x]$ *# We recommend the probabilistic t-conorm*

 # Perform optimization of the graph layout
$Y \leftarrow$ SpectralEmbedding(top-rep, d)
$Y \leftarrow$ OptimizeEmbedding(top-rep, Y, min-dist, n-epochs)
return Y

Figure 3.10: UMAP algorithm[10]

They use the KNN classifier to evaluate the performance of different decomposition methods in two different data sets: COIL-20 and PenDigits. UMAP has better and more stable performance in both two data sets compared with t-SNE. (see Figure 3.11).

	k	t-SNE	UMAP	LargeVis	Eigenmaps	PCA
COIL-20	10	**0.934** (± 0.115)	0.921 (± 0.075)	0.888 (± 0.092)	0.629 (± 0.153)	0.667 (± 0.179)
	20	0.901 (± 0.133)	**0.907** (± 0.064)	0.870 (± 0.125)	0.605 (± 0.185)	0.663 (± 0.196)
	40	0.857 (± 0.125)	**0.904** (± 0.056)	0.833 (± 0.106)	0.578 (± 0.159)	0.620 (± 0.230)
	80	0.789 (± 0.118)	**0.899** (± 0.058)	0.803 (± 0.100)	0.565 (± 0.119)	0.531 (± 0.294)
	160	0.609 (± 0.067)	**0.803** (± 0.138)	0.616 (± 0.066)	0.446 (± 0.110)	0.375 (± 0.111)
PenDigits	10	**0.977** (± 0.033)	0.973 (± 0.044)	0.966 (± 0.053)	0.778 (± 0.113)	0.622 (± 0.092)
	20	0.973 (± 0.033)	**0.976** (± 0.035)	0.973 (± 0.044)	0.778 (± 0.116)	0.633 (± 0.082)
	40	0.956 (± 0.064)	0.954 (± 0.060)	**0.959** (± 0.066)	0.778 (± 0.112)	0.636 (± 0.078)
	80	0.948 (± 0.060)	**0.951** (± 0.072)	0.949 (± 0.072)	0.767 (± 0.111)	0.643 (± 0.085)
	160	0.949 (± 0.065)	**0.951** (± 0.085)	0.921 (± 0.085)	0.747 (± 0.108)	0.629 (± 0.107)

Figure 3.11: Comparison between different decomposition methods[10]

Except for their evaluation, the effectiveness of UMAP has been shown in other researches, e.g. Becht et.al[11] apply UMAP in single-cell analysis from the field of cytobiology. Their study also makes a qualitative comparison of UMAP with t-SNE and the other five decomposition tools. The results show that the UMAP tool has the best performance among them. (see Figure 3.12)

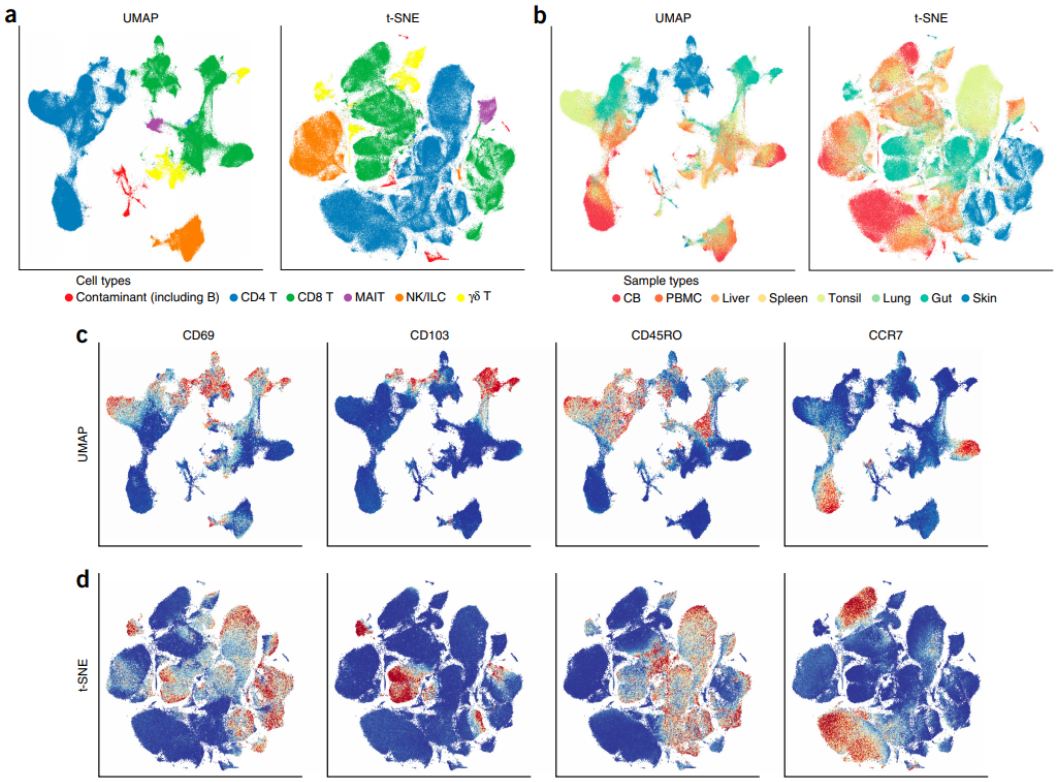


Figure 3.12: Results in the single-cell analysis using four different decomposition methods[11]

We used the $PCA()$ and $UMAP()$ from the libraries of *sklearn* and *umdp*. The values of the hyper-parameters in both functions are set to 3 since we need the three-dimensional representation of the original text vectors. Partial code details can be found in appendix **??**.

3.2.3 ANN Model Implementation

All of our implementation of the ANN models is finished through Keras. It is an easy-to-use neural network API written in Python,[61][62]. Before we introduce the implementation of each model, we will summarise the common functions from Keras, which provides a modular method of building ANN. In our project, we use $Sequential()$, $Dense()$, $Flatten()$, and $Embedding()$ to build the model. $Sequential()$ function can be regarded as the entry function of the entire neural network model, to be more

specific, *Sequential*() function defines that this model is the simplest linear and initiate the whole models which allow the model to accept other types of ANN middle layers. *Dense*() function is the pre-defined fully connected layer from Keras, where provides multiple parameters, including whether using bias vectors. The parameter we will use is *units*, the required parameter in the *Dense*() function and other pre-defined model layer functions. The other parameter we will use is *activation*, which decides what to launch to the next neuron. We have a more detailed introduction in Chapter 2. The optional values for parameter *activation* are accessible for users to transfer the string values, which is the name of those pre-defined activation functions, including softmax, elu, relu, sigmoid, etc. We use *relu* as the activation function in the hidden layer and *sigmoid* in the output layer. We choose *sigmoid* as the output layer activation function because of our project two-class task. As a result, the output of our model should also be binary, and *sigmoid* is a classical binary activation function.
Flattern() is an implementation of the flattening layer in Keras. The Flatten layer is used to "flatten" the input, to transform the multi-dimensional input data into data in a format of a single dimension, which greatly reduces the use of parameters and avoids over-fitting. The parameter we set is *unit*, and its effect is the same as the parameter in the layers to decide the output data dimension.
Embedding() stands for the implementation of the embedding layer. The purpose of embedding is to project high-dimensional, relatively sparse data in each dimension to a relatively low-dimensional one. Each dimension can take the data operation of the real number set. We use this function to process the input data.
The first ANN model we implement is MLP. Due to its concepts, MLP is the simplest ANN model in the four ANN models we implement. All the functions we use have been introduced previously. The model structure is shown in the Figure 3.13 drown by the Flask internal functions.

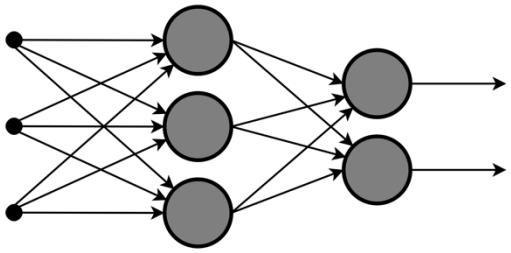


Figure 3.13: MLP model structure

For CNN model, compared with MLP model, the functions that we specifically use are $Dropout()$, $Conv1D()$ and $GlobalMaxPooling1D()$. In $Dropout()$, some neurons will be randomly deleted to train different neural network architectures on different batches, and we use a 0.3 ratio in this function to get the best performance. $Conv1D$ is one of the most crucial layers in CNN, responsible for extracting different input features and passing the result to the next layer. $GlobalMaxPooling1D$ is a kind of pooling layer and method. The meaning of the word "global" is that the size of the sliding window (decide the area to process pooling) equals the size of the whole feature map, and "max" means that the output element of the new feature map is the largest element of the input feature map in the area of a sliding window. The model structure diagram can be reviewed in Figure 3.13.

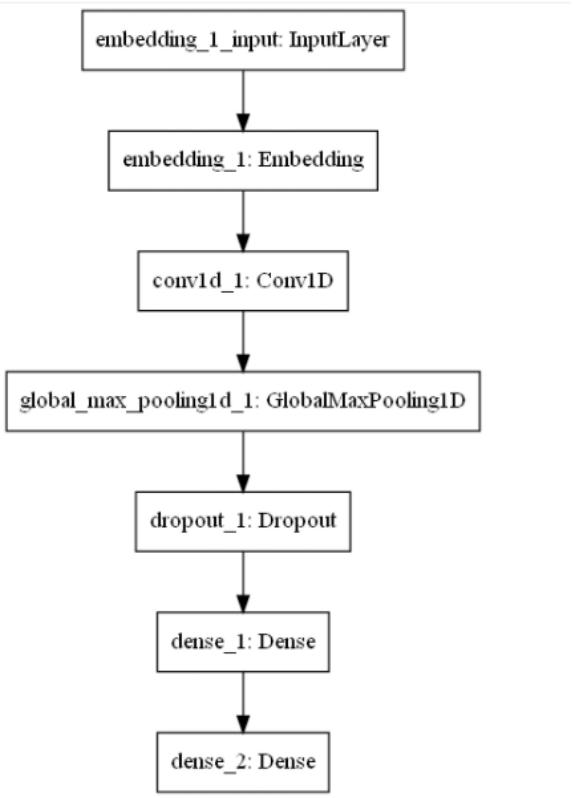


Figure 3.14: CNN model structure

For the RNN model, the only layer that we introduce here is $SimpleRNN()$. $SimpleRNN()$ is a function from Keras that contains a complete RNN layer function. The only parameter we set is $units$. The model structure is shown in Figure 3.14.

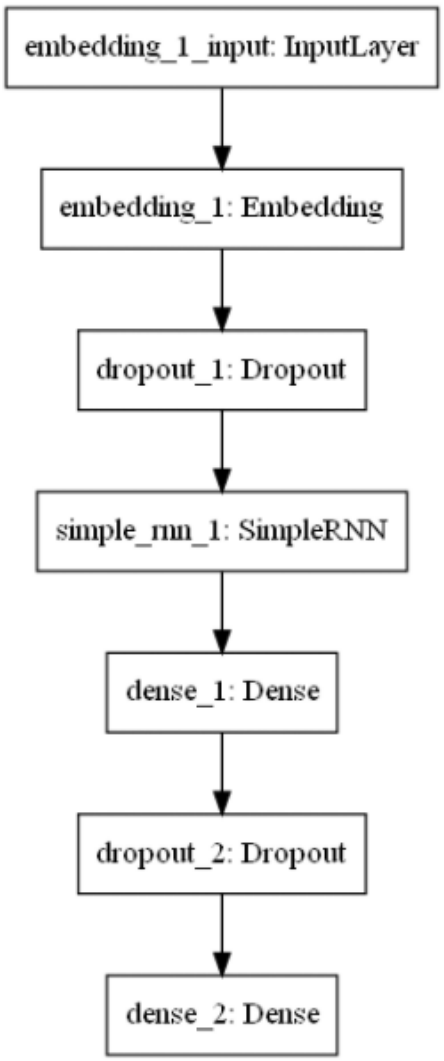


Figure 3.15: RNN model structure

The last model we implement is LSTM, like we introduced in chapter 2, is an upgraded RNN model, so the structure of LSTM is very similar to RNN, and the LSTM structure is implemented by $SLTM()$, in where we use the parameter of *units* only. The model structure of LSTM is shown in Figure 3.15.

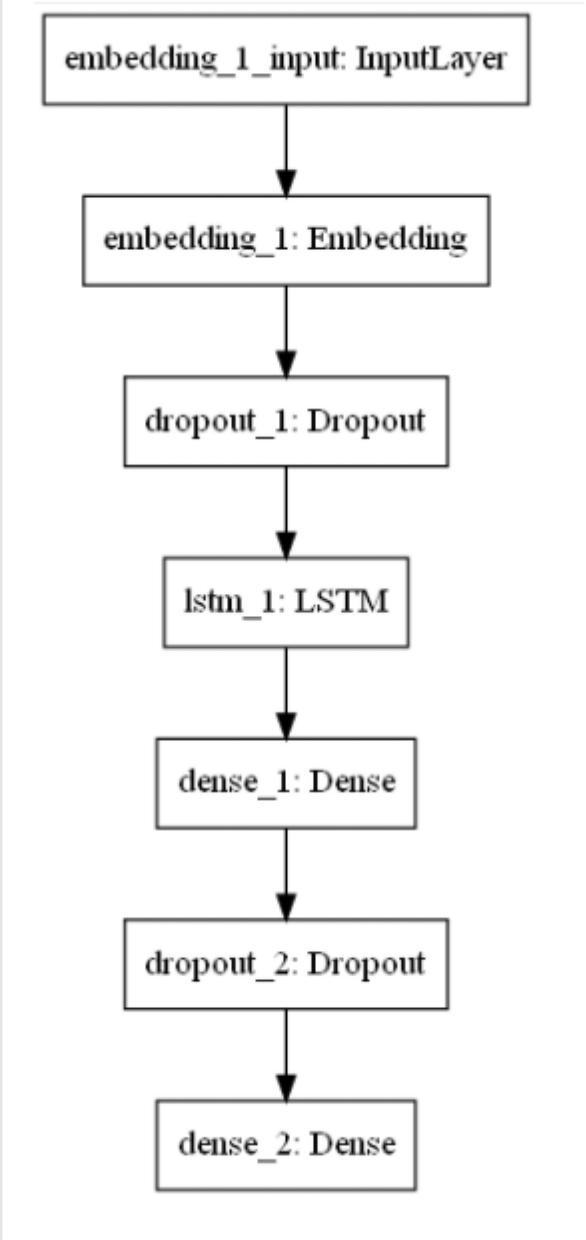


Figure 3.16: LSTM model structure

3.2.4 Flask Route Building

We use Flask to build the back-end route to provide the API for users to use this project's functionalities. Flask is a lightweight web application framework written in

Python based on Werkzeug and WSGI[63][64], we did not choose other frameworks like Django since they can provide more features indeed, but the standard Django project framework established by scaffolding commands will create more pressure for the server compared with Flask project.
We design four endpoints in Flask in total. The first endpoint is "/get_vecs", which provides the service for users to get the generated vectors from the back-end to be shown in the "Visualization" section. The second endpoint is "/get_vec_detail", where receiving the information of the clicked point in the "Visualization" section as parameters and giving the "processed_review", "original_review" and "sentiment" as the response. The third endpoint is "/get_corpus_stat", we count the number of comments in different length ranges and return the result of counting as the response. The fourth endpoint is "/get_ann_model". We provide the interface to receive the parameters of the hyper-parameters selected by the users and be responsible for training new ANN models and returning the data of training history to be presented in the front-end.
Besides, we adopt a project development model of separation of front-end back-end, Cross-Origin Resources Sharing (CORS) must be considered and solved. When the browser sends an Ajax request, it only receives the data resources that the same origin server responds to. Only the protocol, origin name, and port are all the same can be considered under the same domain. If one of the three conditions is inconsistent, it is not considered the same origin, cross-origin. Flask has provided the $CORS()$ function to implement CORS configuration, we use the default configuration supported in $CORS()$, the method to use this function is to simply wrap the instance of $Flask()$ before calling its $run()$ function.

3.2.5 Front-end Design

The other main point of this project, besides the ANN model implementation, is the design of the front-end page.
Considering the services that this project provides, I decided to follow the principles of Single Page Application (SPA). However, it is generally believed that there is no clear and authoritative definition of SPA, but in work from Mesbah and Van Deursen[65], in where they give some critical features of SPA:

- Components: the weight of term i in document j.

- Updated/replaced: the frequency of term i in document j

- Reloaded: number of documents in the data set.

After establishing the broad design principles, we decide to use React Hooks to build the components need in the front-end pages, compared with classical React, where create a class inherited from React. Component to design own class, we could directly create our component class through custom functions instead of class inheritance. Easier to reuse code and effectively reducing code complexity are two main advantages of using React Hooks[66][67]. However, when facing with asynchronous requirements, it is defective compared to class inheritance. While in our project, there is not a large number of asynchronous requests, we think this defect is negligible.
Following the rules of SAP and applying React Hooks and ideas from SAP, we finish the front-end UI design. (see Figure 3.4)

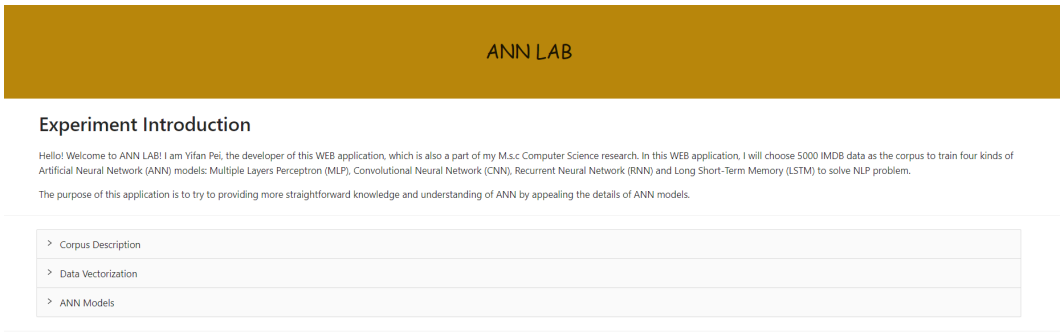


Figure 3.17: Project UI implementation

On the initial page, we put a brief introduction of this project, which includes the purpose of our research, the developer's name, and a summary of the project.
In Figure 3.4, we divide the services we provide into three sections, and what is included in the first part is showed in Figure 3.5.

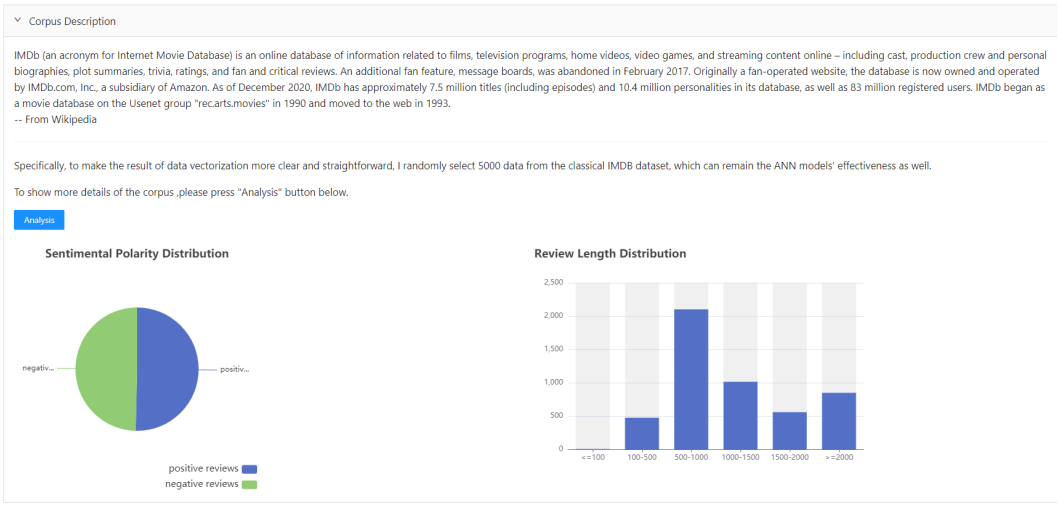


Figure 3.18: UI of section of "corpus description"

In the part of "corpus description", we firstly quote a description of the IMDB database from Wikipedia to let the users have a preliminary understanding of the source and content of the data set. Then we design two graphs: a pie chart of sentiment polarity distribution and a bar chart of reviews lengths in different ranges to describe the data distribution in the data set we will use. The purpose of sentiment polarity is to prove to users that the data distribution in the data set used by our project is balanced. At the same time, this icon can also indirectly prove the effectiveness of the model we implemented. On the one hand, we want to show more details of the data set for the other graph. On the other hand, we also believe using this chart could provide a chance for users to build a connection between the length of the reviews and the performance of the ANN models, thereby gain more knowledge about the ANN models.

For the implementation of this section, we import components from *Echarts* to draw these two graphs. We use $useContext$ and $useEffect$ from React Hooks to manage the status, $useContext$ could receive the variables defined by the developers and turns them into global variables through $< ContextProvider >$ tag, used in the JS file of the project component configuration and wrap the root node, in order to share the global variables with all components in the project. Nevertheless, it does not necessarily need to wrap the root node, wrap components at any level so that variables can be shared with that component to achieve the effect of using global variables. React also provides

Redux to manage the whole status, but based on the scale and needs of our project data transmission and exchange. We believe using *Redux* will only unnecessarily increase the complexity of the project.
To interact with the "corpus description" section, the users need to click the button of "Analysis", and requests will be created and sent by *Axios* to $useEffect$ function to operate the settings of two graphs provided by $useContext$.
The second section, UI design, can be seen in Figure 3.6.

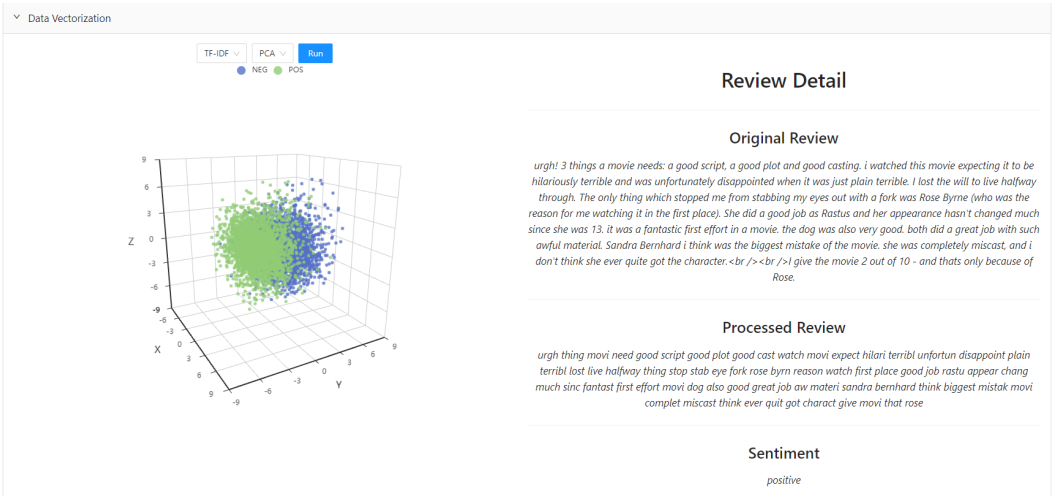


Figure 3.19: UI of section of "visualization"

In this part, we recommend that the users use multiple methods to see the visualization results, which contains two operations: vectorization and decomposition. To save the time that users need to wait for the response, we store the visualization result in the format of ".npy" to give the response as soon as possible.
In the scatter chart, each point represents the review in the data set. We also allow the users to click the point. As a result, three pieces of data will be shown in the correct blank area: original review preprocessed review and the sentiment. Based on the decomposition theory, the specific values on the three coordinate axes are not very meaningful. The purpose of drawing the scatter chart is to show the distance of the points in 3D space. Combining with the comment data displayed in the blank area on the right, the users can have a deeper understanding of review data in the same sentiment polarity. e.g. Review data A is positive, and the review data B is negative, and

here is a positive review data C. However, review A and B are both positive reviews but comparing the relative distance between point C and point A and point B, users can discover more information through the comparison.
In the last section – "ANN Models", which we are directly inspired by "ANN Playground" by Tensorflow, we allow users to learn about the ANN models and manipulate those models. The UI design is showed in Figure 3.7.

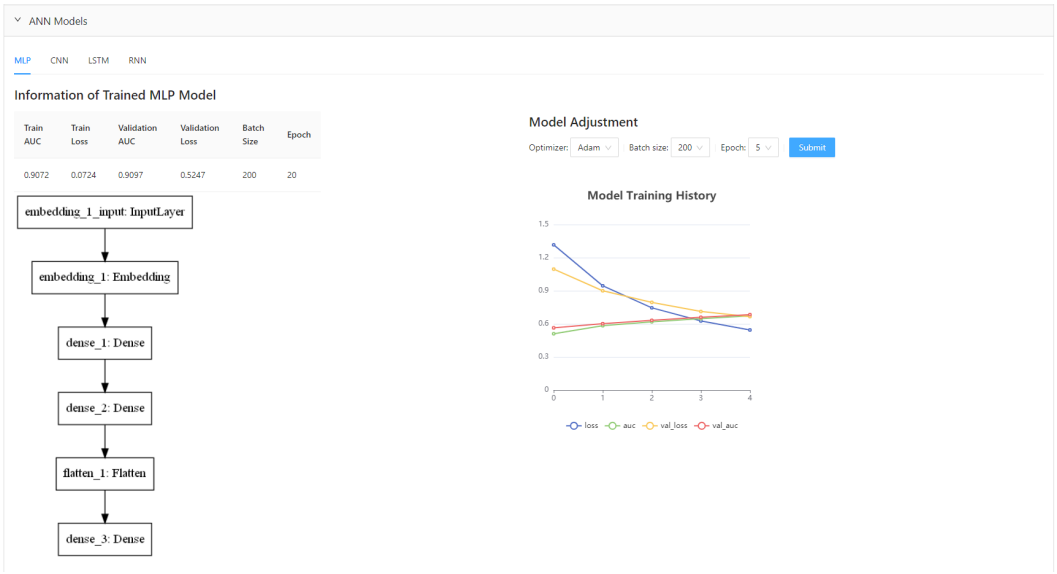


Figure 3.20: UI of section of "visualization"

In this section, we provide four options for users to choose and each option represents an ANN model: MLP, CNN, RNN and LSTM. Once the user clicks the panel, a sub-area will appear below, and on the left side, there will be a structure diagram of the model selected, which includes the name of each layer and the data flow through the model. Moreover, on the other side, the interface of models manipulation is provided, the users can tune the hyper-parameters of the ANN models, including the choice of "optimizer", which is the optimal solution of the model, and the primary purpose is to get the smallest possible value in the loss function, the number of "epoch", which stands for the number of times to train using the entire training data, and the number of "batch size", which is the sample size used in one iteration. The changeable hyper-parameters and optional values are shown in Table 3.3.

Table 3.3: Optional Values of Hyper-parameters from ANN Models

	Epoch	Batch Size	Optimizer
Options	1, 5, 10, 15, 20	100, 200, 400, 800, 1000	SGD, RMSprop, Adam

When the users decide on the hyper-parameters, they can click the "Submit" button, the request will be sent to the Flask back-end to re-compile and re-train the pre-trained ANN models stored in the format of ".h5". After training time as short as a few seconds or as long as a few minutes, the back-end will transfer the "training history" into the front-end line graph settings and update the line graph to show the variation during training.

Chapter 4

Evaluation

This chapter will state the metrics that we choose to evaluate our ANN models to ensure that the models we provide for the users are effective and will not cause any misunderstanding.
During the evaluation, based on the data set that we choose, a 5000 sub-data-set of IMDB, we used the *train_test_split*() function from sklearn to divide the 5000 data into the training set and test set at a rate of 0.3. As a result, 3500 data will be used in the training of the ANN model, and 1500 data will be used in test and validation.

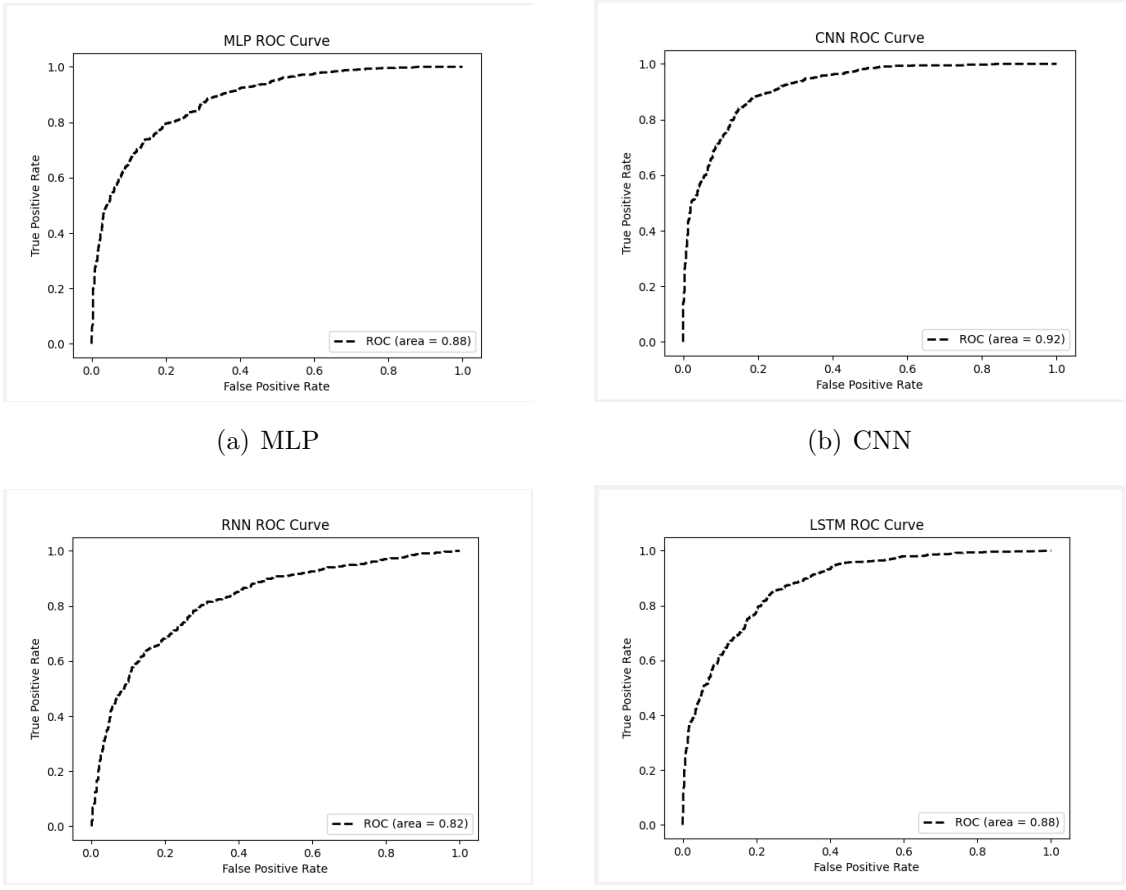
4.1 Metrics for ANN Models

This section will present three criteria applied in our project's evaluation stage: confusion matrix, ROC and AUC and ANN models training history. These criteria are helpful for us to appraise the performance of our models.

4.1.1 ROC and AUC

ROC (AUC) is a commonly used criterion in classification tasks to evaluate the algorithm performance. It is coming from the concepts of the confusion matrix. In a particular data set and classifier, the researchers can only get one confusion matrix or a group of TP, FP, TN and FN, to get continuous data to draw ROC, the data used in ROC coming from the probability prediction of classifier but not actual category classification prediction.

In our project, there are four kinds of ANN models. To draw the ROC, callable functions *roc_curve*() and *auc*() are accessible. (see Figure 4.1)



(a) MLP

(b) CNN

(c) RNN

(d) LSTM

Figure 4.1: ROC and AUC of ANN models

In Figure 4.1, it can be seen that the ROC is nearly the same in all ANN models, but the most significant value of AUC is from the CNN model, which is 0.92.

4.1.2 Confusion Matrix

The confusion matrix describes the classification accuracy of a classifier. The word "confusion" also vividly expresses the confusion that the classifier may cause when facing multiple classifications. It is a performance measure for machine learning clas-

sification problems, where the output can be two or more classes. The table contains four different combinations of predicted and actual values. The actual value in the confusion matrix is also called "target", "reference", "actual". Correspondingly, the predicted value is called "model", "prediction", "predicted". The binary classification is identified as Positive and Negative, sometimes also identified as "Normal/Abnormal", "Accept/Reject", or more simply as "Yes/No."
The tables of confusion matrices of four ANN models are presented from Table 4.1 to 4.4.

Table 4.1: Table of Confusion Matrix of MLP Model

Predict / True	Negative	Positive
Negative	600	160
Positive	130	610

As showed in the Figure 4.1, the True Positive Rate (TPR) is 82.43%, and the False Positive Rate (FPR) is 21.05%.

Table 4.2: Table of confusion matrix of CNN model

Predict / True	Negative	Positive
Negative	610	150
Positive	80	660

As showed in Figure 4.2, the True Positive Rate (TPR) is 89.19%, and the False Positive Rate (FPR) is 19.74%.

Table 4.3: Table of Confusion Matrix of RNN Model

Predict / True	Negative	Positive
Negative	570	180
Positive	160	590

As showed in Figure 4.3, the True Positive Rate (TPR) is 78.67%, and the False Positive Rate (FPR) is 34.00%.

Table 4.4: Table of confusion matrix of LSTM model

Predict / True	Negative	Positive
Negative	570	180
Positive	120	640

As shown in Figure 4.4, the True Positive Rate (TPR) is 84.21%, and the False Positive Rate (FPR) is 24.00%.
Compared to all the statistics in the tables and the calculated TPR and FPR, the model with the best quality in the ROC metrics is RNN, with the value of TPR is 89.19% and the value of FPR is 19.74%.

4.1.3 Training History

There has provided callable functions to get the history from the ANN models training progress in Keras, combing with Python library Matplotlib, the changes in various indicators during training.
Based on the internal structure of "history" returned (see Figure 4.2), the value of the x-axis and y-axis can be decided.

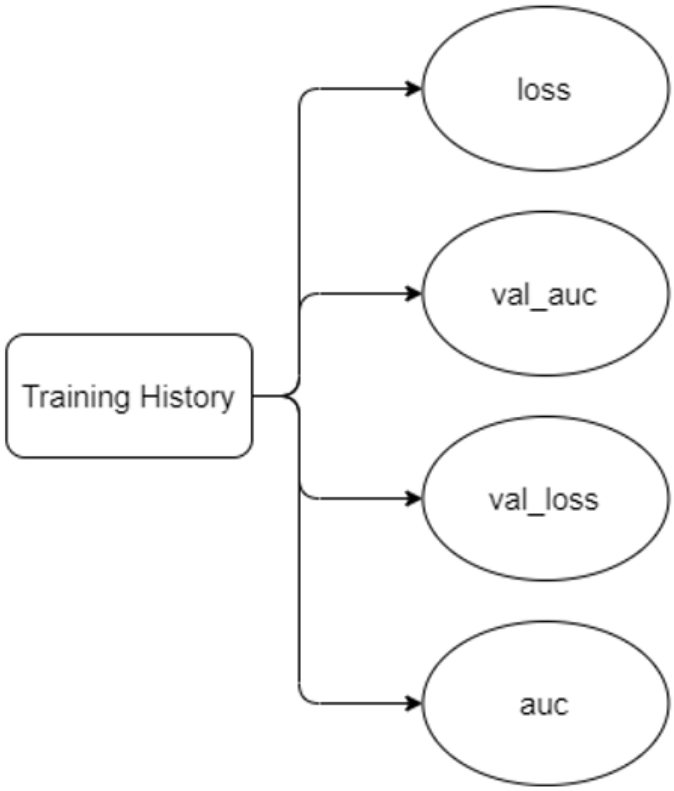
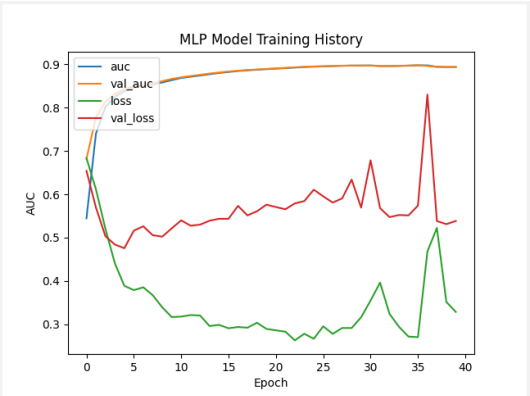
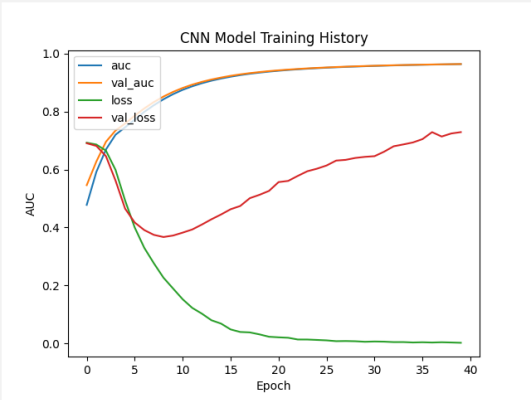


Figure 4.2: A screenshot of the internal structure of partial training history

To draw the training history line graphs for four pre-trained ANN models, we use the value of "epoch" in the x-axis and the value of "ROC and AUC" in the y-axis. Thus, the line graph is showed in Figure 4.3.



(a) MLP



(b) CNN

(c) RNN

(d) LSTM

Figure 4.3: Training History of ANN Models

In the sub-figures in Figure 4.2, it can be seen that after 20 epochs, the values of "val_auc" and "auc" has reached a relatively high level (almost 90%). However, it can be seen that after 20 epochs, the values of "loss" and "val_loss" even starts to increase. This situation is undeniable in the LSTM model. But the "auc" and "val_auc" show that the training of the four ANN models is practical.

The evaluation of the training history of ANN models is also a reference for us to set the range of epoch that we can provide for users to use in adjusting our pre-trained ANN models. We make a trade-off here that set the maximum epoch that users can use is 20.

Chapter 5

Conclusion and Future Work

This chapter will present the conclusion of this dissertation, the limitations of this project, and the resulting possible future work.

5.1 Conclusion

In this project, we designed a Web application, React and React Hooks are used with Ant Design to build the front-end UI. For the back-end side, there are two parts. The first part is responsible for building and training ANN models and other related work, including vectorization and decomposition using Keras, sklearn and Numpy, the second part is the route, which is implemented by Flask, to make the services that we implemented available to the users once we host the project. All the ANN models: MLP, CNN, RNN and LSTM reach the AUC precision around 90%.
For the data set used in this project, we are considering the IMDB data set, which contains the movie-related data, e.g. review and the information of actors directors, but we select 5000 data from a sub-dataset of IMDB, which only contains the movie review and the corresponding sentiment to use as the input data in the ANN models because the original IMDB data set is too large to implement the function that the user can adjust the ANN model and get a response in real-time.

5.2 Limitations

Limited Types of Corpus

In this project so far, only one type of corpus can be used: IMDB data set. However, the performance of the four kinds of ANN models has been proved to be effective. However, it is not sound enough to use only one kind of data set since the users cannot get more knowledge about the data set's influence on the ANN model performance. Therefore, maybe trying different data sets, incredibly different types of a corpus, e.g. long text in sentence-level, documents collection in document-level and even conversation data collected in real-time.

Unclear Visualization of Training Progress

Even though we try to visualize every training detail, but restricted to the programming level, we still cannot gain more information during the ANN model training (e.g. the variation of inner hidden layer matrix) and return the data in real-time, modifying the source code of the functions from Keras and build own classes for it may become a solution.

ANN Training Slowness

Slow ANN training is a severe problem in the future progress since it is intolerable for users to see the training progress when it is processing a much larger corpus, changing the current project architecture, in where the front-end and back-end are separated, into an integrated front-end and back-end structure maybe can be put into consideration.

5.3 Future Work

In addition to the last section of "Limitations", other potential works might be considered in the future.
The first possibility is the type of sentiment polarity from the data set. However, we have mentioned in the last section that replacing data set with longer or shorter text would be a possible work changing binary classification into a multi-class classification

and even design the ANN models which have the adaptive ability towards both binary classification and multi-class classification. It will provide the users with more views when they are using ANN models.

Besides, an extensive user feedback survey before any further work has been done is available to get more ideas. It is a valuable method to get the direction that this project should follow in the future.

Bibliography

[1] R. Feldman, "Techniques and applications for sentiment analysis," *Communications of the ACM*, vol. 56, no. 4, pp. 82–89, 2013.

[2] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[3] N. Fouladgar and K. Främling, "Xai-pt: a brief review of explainable artificial intelligence from practice to theory," *arXiv preprint arXiv:2012.09636*, 2020.

[4] L. Viganò and D. Magazzeni, "Explainable security," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 293–300, IEEE, 2020.

[5] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," *Information processing & management*, vol. 50, no. 1, pp. 104–112, 2014.

[6] S. Vijayarani, M. J. Ilamathi, M. Nithya, *et al.*, "Preprocessing techniques for text mining-an overview," *International Journal of Computer Science & Communication Networks*, vol. 5, no. 1, pp. 7–16, 2015.

[7] I. A. El-Khair, "Effects of stop words elimination for arabic information retrieval: a comparative study," *International Journal of Computing & Information Sciences*, vol. 4, no. 3, pp. 119–133, 2006.

[8] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," *arXiv preprint arXiv:1309.4168*, 2013.

[9] G. Destefanis, M. T. Barge, A. Brugiapaglia, and S. Tassone, "The use of principal component analysis (pca) to characterize beef," *Meat science*, vol. 56, no. 3, pp. 255–259, 2000.

[10] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[11] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell, "Dimensionality reduction for visualizing single-cell data using umap," *Nature biotechnology*, vol. 37, no. 1, pp. 38–44, 2019.

[12] A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (xai)," *IEEE access*, vol. 6, pp. 52138–52160, 2018.

[13] H. C. Lane, M. G. Core, M. Van Lent, S. Solomon, and D. Gomboc, "Explainable artificial intelligence for training and tutoring," tech. rep., UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY CA INST FOR CREATIVE ..., 2005.

[14] J. Suh, S. Yoo, J. Park, S. Y. Cho, M. C. Cho, H. Son, and H. Jeong, "Development and validation of an explainable artificial intelligence-based decision-supporting tool for prostate biopsy," *BJU international*, vol. 126, no. 6, pp. 694–703, 2020.

[15] I. Tenney, J. Wexler, J. Bastings, T. Bolukbasi, A. Coenen, S. Gehrmann, E. Jiang, M. Pushkarna, C. Radebaugh, E. Reif, *et al.*, "The language interpretability tool: Extensible, interactive visualizations and analysis for nlp models," *arXiv preprint arXiv:2008.05122*, 2020.

[16] C. D. Manning and P. Raghavan, "and schutze, h.[2008] introduction to information retrieval," 2008.

[17] J. Hutchins, "The first public demonstration of machine translation: the georgetown-ibm system, 7th january 1954," *http://www. hutchinsweb. me. uk/GU-IBM-2005. pdf*, 2005.

[18] N. Chomsky, "Three models for the description of language," *IRE Transactions on information theory*, vol. 2, no. 3, pp. 113–124, 1956.

[19] M. Lauer, "How much is enough?: Data requirements for statistical nlp," *arXiv preprint cmp-lg/9509001*, 1995.

[20] Y. Goldberg, "Neural network methods for natural language processing," *Synthesis lectures on human language technologies*, vol. 10, no. 1, pp. 1–309, 2017.

[21] S. Poria, E. Cambria, and A. Gelbukh, "Aspect extraction for opinion mining with a deep convolutional neural network," *Knowledge-Based Systems*, vol. 108, pp. 42–49, 2016.

[22] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[23] J. J. Hopfield, "Artificial neural networks," *IEEE Circuits and Devices Magazine*, vol. 4, no. 5, pp. 3–10, 1988.

[24] L. Noriega, "Multilayer perceptron tutorial," *School of Computing. Staffordshire University*, 2005.

[25] L. Hontoria, J. Aguilera, and P. Zufiria, "An application of the multilayer perceptron: solar radiation maps in spain," *Solar energy*, vol. 79, no. 5, pp. 523–530, 2005.

[26] H. Daniels and B. Kamp, "Application of mlp networks to bond rating and house pricing," *Neural Computing & Applications*, vol. 8, no. 3, pp. 226–234, 1999.

[27] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*, pp. 267–285, Springer, 1982.

[28] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[29] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[30] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," *Advances in neural information processing systems*, vol. 27, pp. 1790–1798, 2014.

[31] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[32] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[33] R. Lin, S. Liu, M. Yang, M. Li, M. Zhou, and S. Li, "Hierarchical recurrent neural network for document modeling," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 899–907, 2015.

[34] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[35] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[36] M. Van Lent, W. Fisher, and M. Mancuso, "An explainable artificial intelligence system for small-unit tactical behavior," in *Proceedings of the national conference on artificial intelligence*, pp. 900–907, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[37] F. Emmert-Streib, O. Yli-Harja, and M. Dehmer, "Explainable artificial intelligence and machine learning: A reality rooted perspective," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 6, p. e1368, 2020.

[38] R. Andrews, J. Diederich, and A. B. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowledge-based systems*, vol. 8, no. 6, pp. 373–389, 1995.

[39] A. J. London, "Artificial intelligence and black-box medical decisions: accuracy versus explainability," *Hastings Center Report*, vol. 49, no. 1, pp. 15–21, 2019.

[40] A. Kuppa and N.-A. Le-Khac, "Black box attacks on explainable artificial intelligence (xai) methods in cyber security," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2020.

[41] A. Deeks, "The judicial demand for explainable artificial intelligence," *Columbia Law Review*, vol. 119, no. 7, pp. 1829–1850, 2019.

[42] S. Kannan, V. Gurusamy, S. Vijayarani, J. Ilamathi, M. Nithya, S. Kannan, and V. Gurusamy, "Preprocessing techniques for text mining," *International Journal of Computer Science & Communication Networks*, vol. 5, no. 1, pp. 7–16, 2014.

[43] J. B. Lovins, "Development of a stemming algorithm.," *Mech. Transl. Comput. Linguistics*, vol. 11, no. 1-2, pp. 22–31, 1968.

[44] P. Willett, "The porter stemming algorithm: then and now," *Program*, 2006.

[45] W. Kraaij and R. Pohlmann, "Porter's stemming algorithm for dutch," *Informatiewetenschap*, pp. 167–180, 1994.

[46] N. H. Ali and N. S. Ibrahim, "Porter stemming algorithm for semantic checking," in *Proceedings of 16th international conference on computer and information technology*, pp. 253–258, 2012.

[47] M. V. B. Soares, R. C. Prati, and M. C. Monard, "Improvement on the porter's stemming algorithm for portuguese," *IEEE Latin America Transactions*, vol. 7, no. 4, pp. 472–477, 2009.

[48] C. G. Patil and S. S. Patil, "Use of porter stemming algorithm and svm for emotion extraction from news headlines," *International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCSE)*, vol. 2, no. 7, p. 9, 2013.

[49] V. Balakrishnan and E. Lloyd-Yemoh, "Stemming and lemmatization: a comparison of retrieval performances," 2014.

[50] T. Korenius, J. Laurikkala, K. Järvelin, and M. Juhola, "Stemming and lemmatization in the clustering of finnish text documents," in *Proceedings of the thir-*

teenth ACM international conference on Information and knowledge management, pp. 625–633, 2004.

[51] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[52] W. J. Wilbur and K. Sirotkin, "The automatic identification of stop words," *Journal of information science*, vol. 18, no. 1, pp. 45–55, 1992.

[53] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.

[54] K. S. Jones, "Idf term weighting and ir research lessons," *Journal of documentation*, 2004.

[55] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[56] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, pp. 1188–1196, PMLR, 2014.

[57] D. Kim, D. Seo, S. Cho, and P. Kang, "Multi-co-training for document classification using various document representations: Tf–idf, lda, and doc2vec," *Information Sciences*, vol. 477, pp. 15–29, 2019.

[58] M. Ringnér, "What is principal component analysis?," *Nature biotechnology*, vol. 26, no. 3, pp. 303–304, 2008.

[59] S. Karamizadeh, S. M. Abdullah, A. A. Manaf, M. Zamani, and A. Hooman, "An overview of principal component analysis," *Journal of Signal and Information Processing*, vol. 4, no. 3B, p. 173, 2013.

[60] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[61] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[62] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[63] F. A. Aslam, H. N. Mohammed, J. M. Mohd, M. A. Gulamgaus, and P. Lok, "Efficient way of web development using python and flask," *International Journal of Advanced Research in Computer Science*, vol. 6, no. 2, pp. 54–57, 2015.

[64] M. Grinberg, *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[65] A. Mesbah and A. Van Deursen, "Migrating multi-page web applications to single-page ajax interfaces," in *11th European Conference on Software Maintenance and Reengineering (CSMR'07)*, pp. 181–190, IEEE, 2007.

[66] D. Bugl, *Learn React Hooks: Build and refactor modern React. js applications using Hooks*. Packt Publishing Ltd, 2019.

[67] T. Luojus, "Usability and adaptation of react hooks," 2021.

Appendix

.1 Github Links

The GitHub link for frnt-end:
`https://github.com/Evan-CHN/dissertationProject_fontend`

The GitHub link for back-end:
`https://github.com/Evan-CHN/dissertationPorject_back-end`

.2 Code Pieces from Implementation

.2.1 Overall Text Preprocessing Function

```
def utils_preprocess_text(text, lst_stopwords=None):
    import nltk
    text = processor(text)
    lst_text = text.split()
    lst_text = [word for word in lst_text if word not in
                lst_stopwords]
    ps = nltk.stem.porter.PorterStemmer()
    lst_text = [ps.stem(word) for word in lst_text]
    lem = nltk.stem.wordnet.WordNetLemmatizer()
    lst_text = [lem.lemmatize(word) for word in lst_text]
    text = " ".join(lst_text)
    return text
```

Figure 1: Text preprocessing function

.2.2 Regular Expression Rules in Processing Special Characters

```
def processor(text):
    text = re.sub('<[^>]*>', "", text)
    text = re.sub("[^a-zA-Z]", " ", text)
    emotionicons = re.findall('(?::|;|=)(?:-)?(?:\)|\(|D|P)', text)
    text = (re.sub('[\W]+', " ", text.lower().strip()) + " ".join(emotionicons).replace('-', ''))
    return text
```

Figure 2: Regular expression rules in processing special characters

.2.3 Example of Vectorization Method Implementation

```
def vectorize(train_data, test_data):
    tf = TfidfVectorizer(max_features=1000, ngram_range=(1, 2))
    x_train = tf.fit_transform(train_data['review']).toarray()
    x_test = tf.transform(test_data['review']).toarray()
    from sklearn.preprocessing import scale
    return scale(x_train), scale(x_test)
```

Figure 3: Example of vectorization method implementation

.2.4 PCA Implementation

```
def PCAdecomposition(review_data, sentiment_data, index_data, vectorized_method):
    print('PCAing...')
    pca = PCA(n_components=3)
    pca.fit(review_data)
    review_new = pca.transform(review_data)
    # The remaining code is omitted
```

Figure 4: PCA implementation