AN APPLICATION FOR IDENTIFYING MOVIES FROM PLOT WITH WORD EMBEDDINGS AND DEEP LEARNING

By

Kean Soh Zhe Herng

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

JANUARY 2023

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: AN APPLICATION FOR IDENTIFYING MOVIES FROM PLOT WITH WORD EMBEDDINGS AND DEEP LEARNING

Academic Session: JANUARY 2023

I KEAN SOH ZHE HERNG

declare that I allow this Final Year Project Report to be kept in

Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

[illegible] [illegible]

(Author's signature) (Supervisor's signature)

Address:

12-A, JALAN BPM 9, TAMAN BUKIT

PIATU MUTIARA, BUKIT PIATU,

75150 MELAKA, MALAYSIA

Jasmina Khaw Yen Min

Supervisor's name

Date: 27 APRIL 2023

Date: 28/04/2023

Universiti Tunku Abdul Rahman			
Form Title : **Sample of Submission Sheet for FYP/Dissertation/Thesis**			
Form Number: **FM-IAD-004**	Rev No.: **0**	Effective Date: **21 JUNE 2011**	Page No.: **1 of 1**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 27 APRIL 2023

SUBMISSION OF FINAL YEAR PROJECT

It is hereby certified that KEAN SOH ZHE HERNG (ID No: 19ACB01352) has completed this final year project/ dissertation/ thesis* entitled "**AN APPLICATION FOR IDENTIFYING MOVIES FROM PLOT WITH WORD EMBEDDINGS AND DEEP LEARNING"** under the supervision of DR. JASMINA KHAW YEN MIN (Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

[illegible]

Kean Soh Zhe Herng

DECLARATION OF ORIGINALITY

I declare that this report entitled "**AN APPLICATION FOR IDENTIFYING MOVIES FROM PLOT WITH WORD EMBEDDINGS AND DEEP LEARNING**
" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	[illegible]
Name	:	KEAN SOH ZHE HERNG
Date	:	27 APRIL 2023

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Dr. Jasmina Khaw Yen Min for her patience and continuous support in my endeavours, as well as Ts Dr. Tan Hung Khoon for providing invaluable insight and advice towards the development of this project.

I would also like to thank my coursemates for entertaining me with their shenanigans throughout our final semester.

I wish to dedicate this paper to my family, who have been nothing short of loving and supportive towards my work and studies, for their belief in me is my motivation in completing this paper. Finally, I thank YouTube and its beautiful community, for without the guidance of these genius strangers, this project would be impossible.

ABSTRACT

Natural language processing (NLP) is a field of study in computer science that aims to help computers understand and process human language. Advancements in NLP technology have led to improvements in interactions between humans and computers. Through NLP, the average technology user does not necessarily have to be an expert in computers to “talk” to computers. A common NLP task carried out by computers is multiclass text classification, which allows computers to group documents of similar meaning into one category.

In this paper, a movie identifier from plot which implements the multiclass text classification task mentioned above through a combination of natural language processing and deep learning techniques is proposed to help people who wish to identify movies they have watched in the past but have forgotten their titles. The application can also help people who have heard of bits and pieces of a movie’s plot search for the movie themselves.

The proposed model receives an input of plots from movies extracted from a dataset. Next, preprocessing is performed on the text, such as stemming and lemmatization. Stopwords are removed from the text to discard any words that are not meaningful. The corresponding movie titles of the plots are encoded into integers as targets for the model to predict. The text from the plots is tokenized and encoded into integers as well so that it can be interpreted by the model. As seen in the upcoming parts of this paper, multiple architectures will be reviewed and experimented on. However, most of these architectures follow a similar route in terms of learning features from the text mentioned above, that is transforming the tokens into some sort of embedding layer, subjecting those embeddings through multiple layers in a neural network, and finally classifying the input text and predict the title of the movie referenced in it.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 – Introduction

1.1 Problem Statement and Motivation

In a time when theaters are an affordable and popular place to go to for fun and where infinite selections of films are available to watch on the Internet, it is safe to say that cinema has played a massive role in the lives of many. The increasing accessibility and popularity of online streaming platforms also allow people to watch films from their childhood or a bygone era. However, it is common for many of us to struggle to remember the title of these movies. Some people might have only watched a few movies in their lifetime, or perhaps only caught glimpses of a clip from a movie online. Some even struggle to decide on a movie to watch although presented with a wide array of options.

Problem 1: Inability to recall titles of movies watched in the past

In 2021, Arguello et.al [1] discuss a phenomenon known as *tip of the tongue* (TOT) in describing what it feels like to vaguely remember the details of an item but be unable to recall its name. In the paper, the known items to be identified are movies. An analysis of TOT requests reveals that people tend to forget certain movie titles and attempt to identify them by describing various details of the movie, such as the characters in the movie, the plot summary, the location, etc. Therefore, the proposed application aims to solve the problem of identifying movies that are just at the users' tips of their tongues.

Problem 2: Indecisiveness in picking a movie to watch

Restrictions due to the COVID-19 pandemic have caused theatergoers to abandon their plans for the better part of two years. Typically, people will not have to meticulously decide which movie they would like to watch – the easiest option is to pick the potential blockbuster title. However, within the confines of home and a multitude of selections on the Internet through streaming services, the choices can be overwhelming and cause people to be indecisive when it comes to picking a movie to watch. Hence, the application also serves as a means to help people decide on a movie to watch based on plot elements they are interested in.

1.2 Project Scope

The end product of the development based on this proposal would be a computer application to help users identify movies based on movie plot descriptions. The application would be developed with the Python language on Google Colab.

The application utilizes some popular machine learning libraries such as Pandas, Numpy, Scikit-Learn, NLTK, etc. Google's Tensorflow is used to build the models in this project, specifically by implementing the Keras library. Algorithms such as GloVe and Word2Vec are also experimented with to create word embeddings for the plot texts. Lastly, Qt Designer is used to develop the GUI for the application.

The deep learning models in this project receive input from a dataset of 334 movie titles with around 5 or 6 plots/summaries each. The movie plots are scraped from the "top-rated" and "most popular" pages on IMDB. Preprocessing steps such as stemming, lemmatization, and stopword removal are taken to trim down the meaningful representations of words in the corpus. The text is then tokenized into integers since letters cannot be interpreted by the deep learning model.

Next, a word embedding layer is used to transform the tokens into a high-dimensional vector representation. Many algorithms can be employed to perform the embedding, such as the algorithms mentioned above as well as Keras' default embedding function.

The final layer of the model is usually a dense layer of 334 neurons with a softmax activation function. The number of neurons corresponds to the number of movies in the dataset. Based on the probabilities acquired from each neuron in the final layer, the prediction from the model can be obtained by observing the corresponding index of the neuron with the highest probability.

1.3 Project Objectives

This application aims to help its users identify the titles of movies based on plot descriptions by utilizing natural language processing and deep learning techniques.

Therefore, the objectives of this project are :

1. To identify the title of movies forgotten by the user through the description of plot and dialogue.
2. To recommend a movie to the user based on plot elements, limited only to movies that exist in the data set.

1.4 Impact, Significance and Contribution

Films are one of the most significant forms of art in modern history. With the growing scale of the film industry and the increasing accessibility of movies through the Internet, as well as blockbuster titles such as the two most recent Avengers movies which have become a worldwide phenomenon, cinema has penetrated the lives of many. In daily conversation, it is not uncommon for the subject of movies to be brought up. Through this application, people can quickly search for the title of a movie they might have watched in the past but forgot its name.

Furthermore, despite having large databases of information on almost every movie on the Internet, there is yet to be a mainstream application or website specifically used to identify forgotten movies. This application aims to be unique in the sense that it can stand out and look and feel like more than just a search engine. A main source of inspiration for the design of this application is Akinator [3], which guesses characters instead of movies.

Aside from that, this application would implement various techniques in its development such as natural language processing and deep learning, instead of just performing simple keyword querying on a large database. It is hoped that the model developed will be able to understand the correlation between the input by the user and the plots in the dataset on a contextual scale.

1.5 Report Organization

This report consists of seven chapters. The first and current chapter serves as an introduction to this project as well as explains the main ideas behind its development. The second chapter will consist of a review of all the technologies and literature related to the studies made towards the development of the models and algorithms used in this project. The third chapter will discuss the concepts behind the techniques used in this project as well as provide an overview of the model architecture. The fourth chapter will instead focus on the design of the model based on the concepts discussed in the third chapter of this report. The fifth chapter will instead demonstrate how the designs discussed in the fourth chapter are implemented. The sixth chapter will present the evaluation of the model as well as discuss the performance of the model. Finally, in the seventh chapter, a conclusion and recommendations for future improvements will be given.

CHAPTER 2 – Literature Review

This section will outline some existing techniques and models for multiclass text classification relating to the ones employed in the proposed movie title prediction application. In a departure from the first part of this project, several deep learning architectures and techniques such as word embeddings will be reviewed instead of NLP concepts and machine learning techniques. Then, comparisons will be made between the reviewed architectures and word embedding techniques.

2.1 Word Embeddings

Unlike TF-IDF vector representations of words used in the first part of this project, a more accurate technique used to represent words in a way that can be understood by the deep learning model is to generate word embeddings. Word embeddings represent words in a high-dimensional space and do not solely rely on the frequency of the appearances of words in a corpus. This allows the model to better understand the relationships between each word in the corpus as well as interpret the elements in the text, such as understanding references to objects, characters, places, etc. By feeding word embeddings instead of frequencies of each word as input to the deep learning model, the model is able to process and "understand" the text with more depth.

2.1.1 Word2Vec

It is discovered by Mikolov et.al [4] that relationships between similar words can be represented in a vector space, just like regular numerical values. Words in a corpus with similar meanings will cluster together in that vector space. For example, "Kampar" is more likely to have similar vectors with "Kuala Lumpur", instead of, say, "coffee", due to the former two being both names of cities. Not only that but word vectors can also be further clustered together through different attributes. For example, "Black Sabbath" will be more likely to be clustered with "Metallica" instead of "One Direction". All three are names of bands, but the former two are heavy metal bands while the latter is a pop group. Therefore, this implies that a basic understanding of the world through word vectors can be attained through mathematical operations.

With this in mind, the Word2Vec model utilizes this concept to generate word embeddings based on two algorithms related to the distributional hypothesis mentioned above, which states that words with similar meanings or contexts are clustered together in a corpus. These two algorithms are Skip-gram and continuous-bag-of-words (CBOW).

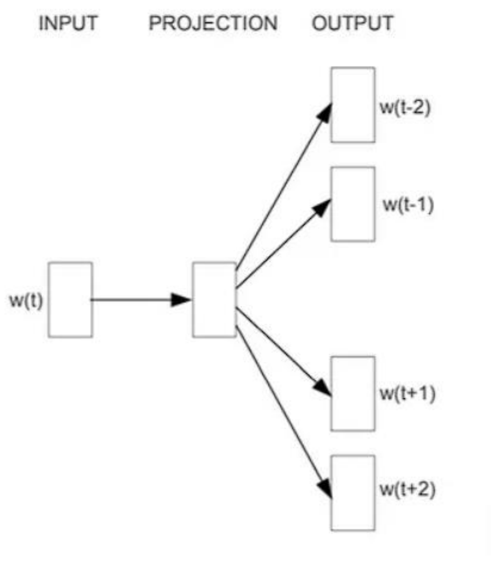


Figure 2.1.1 – Illustration of the skip-gram model's ability to predict surrounding words given a target word

A powerful feature of the skip-gram model is its ability to "read" a sentence and understand its context. It can predict the surrounding words given a target word, by understanding the contexts of words in the corpus. After sampling the words, the model converts all the words in its corpus into a probability distribution and uses the softmax activation function to obtain the words with the highest probability to appear. This means that the skip-gram model has a comprehension of grammar and vocabulary.

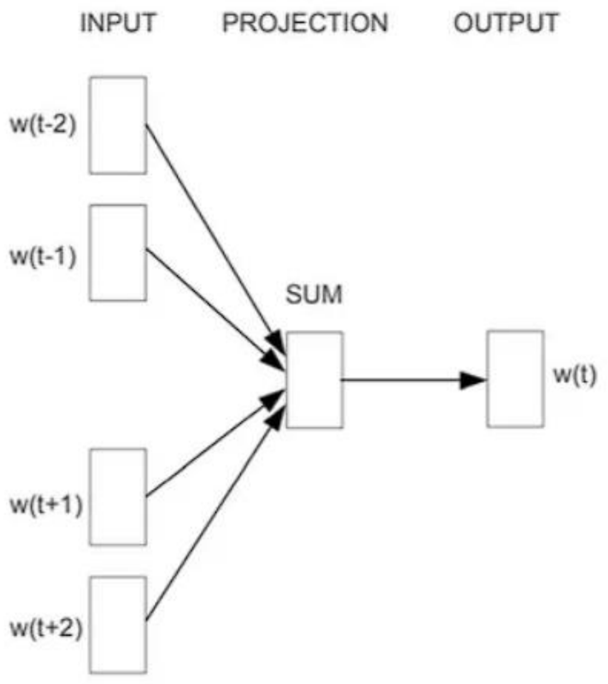


Figure 2.1.2 – Illustration of the CBOW model's ability to predict a target word based on context

On the other hand, the CBOW model can be seen as a sibling of the skip-gram model, in the sense that they have similar concepts such as understanding the relationships between words through context. However, contrary to the skip-gram model, the CBOW model predicts a target word based on a window of context words.

Obviously, to make the prediction of the target word more accurate, a larger window has to be set for the CBOW model to "read" through. This will allow the model to sample more words and in turn, understand the context of the overall sentence better.

As all-encompassing as the two models above may seem in representing the distribution of words, it does come with limitations. Referring to the previous analogy on the grouping of different bands, the very nature of the name “Black Sabbath” is difficult to classify due to a possibility of a lack of context surrounding the words [5] Similarly, Word2Vec might have trouble distinguishing between homonyms, or words that have more than one meaning. In the context of our movie predicting application, a reference to the villain Scorpion in the Spider-Man series might not be accurate as the Word2Vec model might think of it as referring to the actual arachnid instead of the movie character.

2.1.2 GloVe

Global Vectors for Word Representation (GloVe) [6] is another technique used to generate word embeddings. Instead of relying on a small, fixed-length window of words to understand the context of words in a corpus, GloVe generates a co-occurrence matrix for the words in the corpus. The co-occurrence matrix shows how often a word appears alongside every other word in the corpus Essentially, GloVe observes the statistics of the appearances of words in a corpus to model the relationship between them. Theoretically, the more two words appear together, the more likely that they are related in some way.

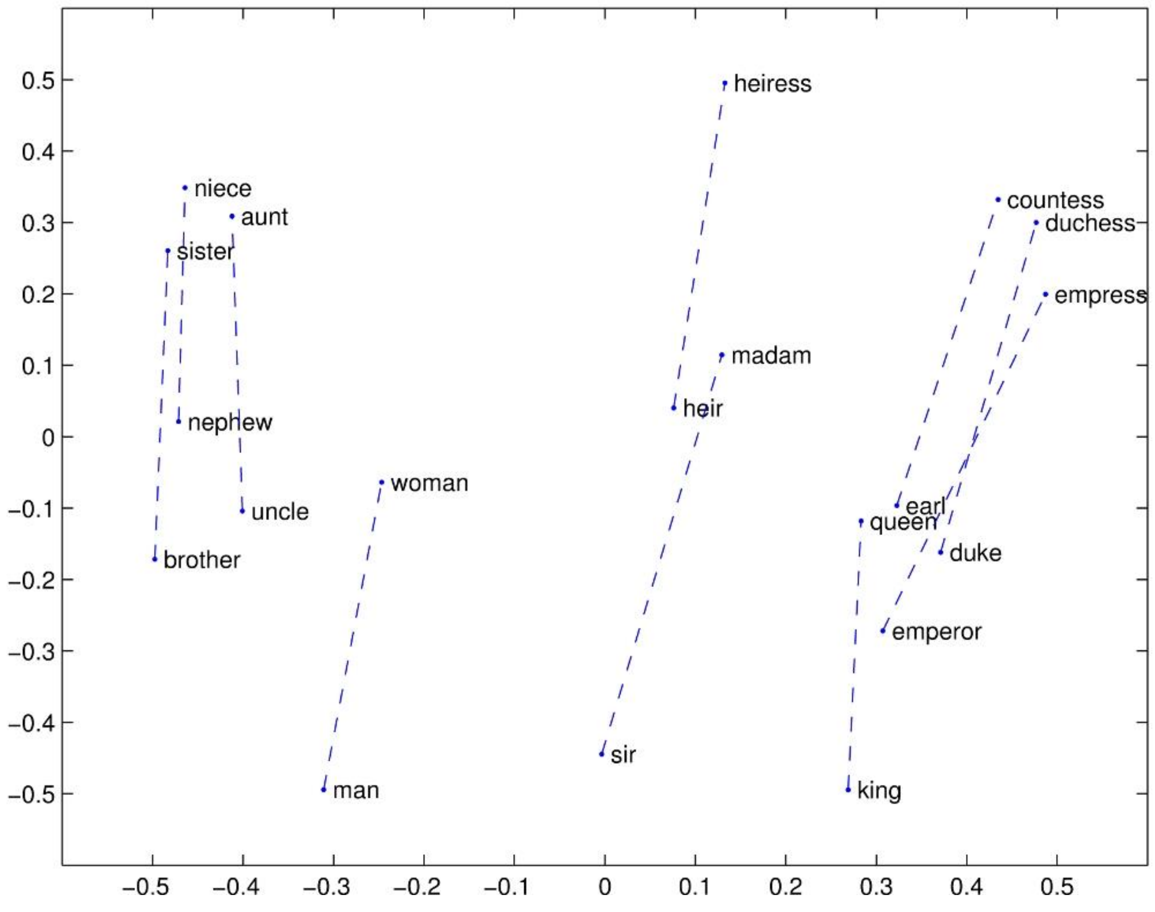


Figure 2.1.3 – Connections made between pairs of words in a corpus by GloVe

However, just by merely comparing the frequencies of each word pair in a corpus is not entirely accurate to extract meaningful representations. For instance, words that commonly appear such as “that” and “is” realistically do not impact the way we perceive the context of words as much as word pairs that occur more infrequently. Therefore, a method called weighted least squares regression is applied to the co-occurrence matrix to bump up the weights of rare but important word pairs. This is similar yet more efficient than the TF-IDF representations mentioned in the first part of this project, in the sense that it does not only focus on the frequency of words but rather also takes into account the importance of each word.

In comparison to the Word2Vec model, GloVe builds connections between each word in the corpus by looking at the entire dataset, whereas Word2Vec only looks at a fixed window of words to infer context. Naturally, this also means that the time needed to train a Word2Vec model is much shorter than GloVe.

2.1.3 Comparison of word embedding techniques

Table 2.1.1 Comparison of word embedding techniques

Technique	Advantages	Disadvantages
Word2Vec	- Has two models, skip-gram and CBOW, able to handle both predicting a series of words given a target word, or vice versa. - Demonstrates much faster training time due to usage of a window of fixed length to "read" from.	- Due to only "reading" from a fixed window, it is difficult for it to understand the context and relationship between words in terms of the entire corpus. - Struggles with distinguishing between homonyms, for example, "bank" as in "river bank" or as in the financial institution.
GloVe	- Handles imbalanced word pair frequencies well, especially in words that do not contribute meaningful representation despite occurring many times. - Looks at the entire dataset to construct co-occurrence matrix, ensuring that the representation of words encompasses the entire corpus.	- Time taken to train a GloVe model from scratch is longer. - Requires a large amount of data to generate accurate word vector representations.

2.2 Model architectures

This section will review some neural network architectures that are suitable for this multiclass text classification task. These models share similar abilities as all of them use numerous layers to learn features from the text given but are also unique in the sense that they have different techniques to achieve the extraction of features in their respective layers.

2.2.1 Convolutional Neural Networks (CNN)

CNNs are more widely used in the field of deep learning for image classification tasks, but they can also be applied to text classification. In image classification, a CNN works by convolving a filter over an image to extract features such as shape, size, and location of said feature within the image. Similarly, in text classification, the convolution is performed over a sequence of text, and each filter applied learns a set of features from it. Specifically, CNNs can learn spatial features from the text, such as the distances of words from one another within the corpus. Relating to the concept behind Word2Vec earlier, CNN can leverage the fact that similar words are likely grouped together and thus is able to deduce semantic similarity from the text.

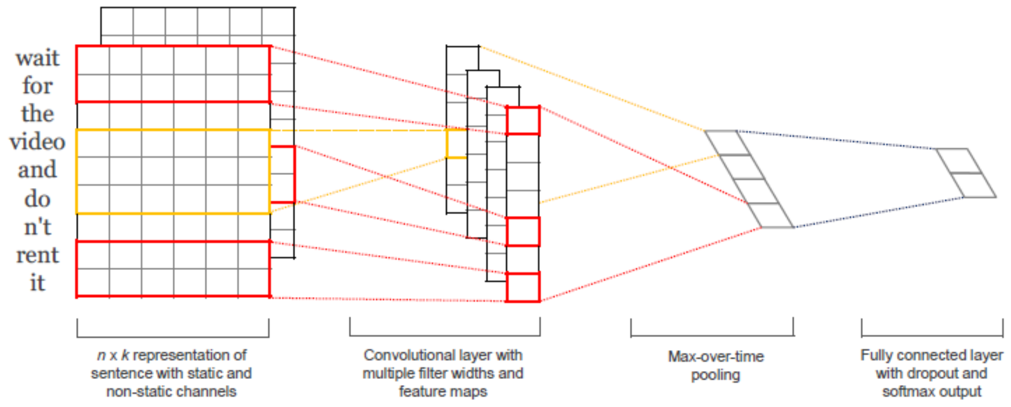

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 2.2.1 – Visualization of CNN feature extraction from text

Based on the figure above from Kim's paper [7], it is observed that the convolution process over text is somewhat similar to that of images. Firstly, a filter with a fixed length is set to convolve over a specific number of words to extract one feature. After the filter convolves over the entire text, all the features are concatenated to generate a feature map. Then, a max pooling operation is applied to the feature map to acquire the feature with the highest value, which corresponds to the most important feature. These features are not the same as individual words, but instead, patterns within the text that are essentially a "compressed" representation of it. With these learned features, CNNs can capture the essence of the text.

2.2.2 Recurrent Neural Networks (RNN)

Instead of "reading" words one filter at a time like CNNs at the risk of discarding information such as context that spans more than the length of the filter, RNNs attempt to understand that context by not only "reading" one token at a time and understanding each one, but also infuse additional information within its "memory" by computing each token with respect to the tokens before. By keeping the "memory" of all the tokens that have come before, RNNs are able to capture the meaning of long sentences.

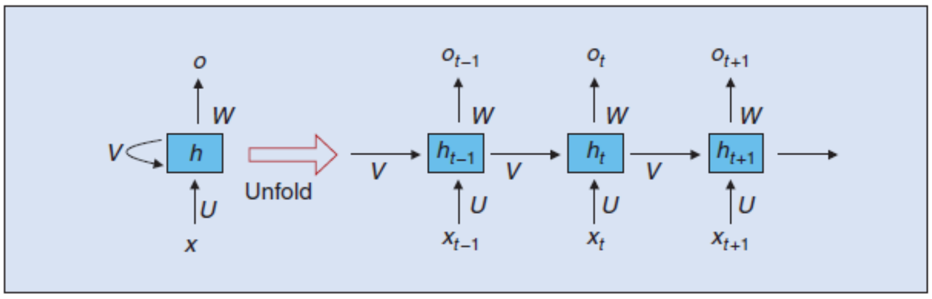


Figure 2.2.2 – Visualization of an RNN network

According to Young et.al [8], an RNN comprehends a sequence of text one token at a time, or in one time step. At each time step, the next token in the sequence is fed into the network, and its hidden state is calculated and updated. This hidden state represents the "memory" of the RNN, and is used to capture the context of a sequence of text, as mentioned earlier. The hidden state is accumulated until the entire sequence is input into the network, and we end up with a dense representation of the sequence.

Another sibling of the RNN, named the Long-Short-Term-Memory (LSTM) network, improves upon the default RNN architecture. Instead of remembering all of the tokens in the sentence, which is a waste of "memory" since not all words in a sentence provide meaningful representation towards a sequence, the LSTM strategically chooses which tokens to drop from its "memory", such as words that do not reappear in a long time after a certain time step.

2.2.3 Bidirectional Encoder Representations from Transformers (BERT)

BERT is an extension of the transformer architecture, proposed by Vaswani et.al [9], which uses a technique called self-attention, which, as the name implies, is able to focus on a specific part in a sequence of text to extract high-level features. These self-attention modules can be multi-headed, which means the transformer is able to focus on many parts of the text at once. This, in turn, allows the transformer to capture long-term dependencies and learn the context of a text sequence more efficiently. Its main components are the encoder and decoder. The encoder receives the input as a sequence of text, and uses the self-attention mechanism mentioned previously to generate hidden representations of the tokens, or in the case of this project, word embeddings, similar to previous subtopics. Once the transformer model has learned the "gist" of the text, it uses the decoder unit to generate another sequence of text, acting like a "paraphraser". However, since this project does not involve text generation, only the encoder part of the architecture is needed.

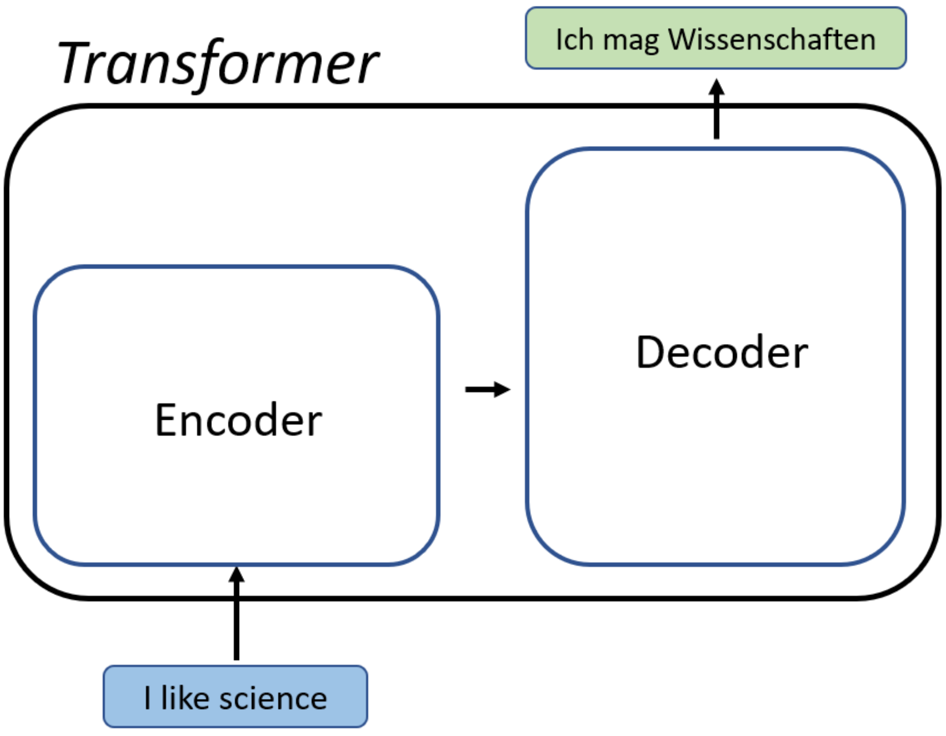


Figure 2.2.3 – An abstract representation of the transformer architecture

Building upon the transformer architecture above, BERT is developed by Devlin et.al [10] as a way to encode tokens in text in a high dimension representation. Its main innovations are that it is not only able to understand context based on previous tokens when encoding, but also it is able to peep into the future, or tokens that occur later in the text sequence (hence the term bidirectional). It also uses a mask during training, which intentionally obscures certain tokens during encoding in such a way that prevents the model from "cheating" by "peering" into the future to learn the actual tokens. This encourages it to focus more on the other parts of the text to obtain the context and discourages it from brute-force memorizing the entire sequence altogether.

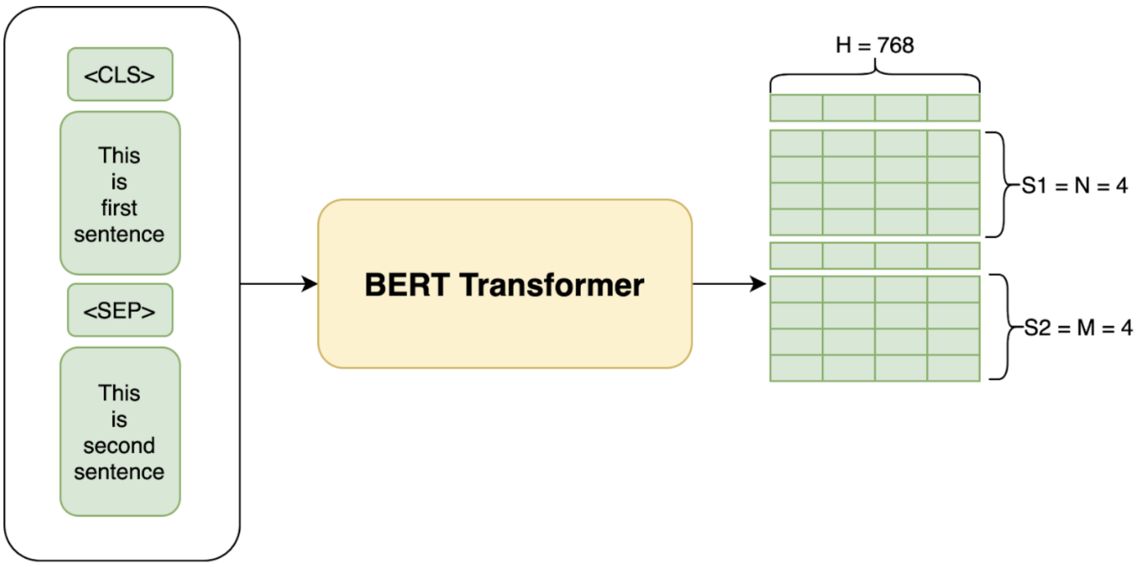


Figure 2.2.4 – A high level view of the BERT transformer

From the diagram above, it is observed that the inputs to a BERT transformer are sentences, and the outputs consist of a matrix of (number of words, 768). This means that for every word received as input to BERT, a vector of size 768 will be generated to represent that particular word in relation to every other word in the corpus.

2.2.4 Comparison of neural network architectures

Table 2.2.1 Comparison of neural network architectures

Architecture	Advantages	Disadvantages
CNN	- Able to learn features from text just from convolving with a small filter - Quick to train	- Unable to capture context on a larger scale, patterns learned are limited to only the size of the filter used
RNN	- Able to learn the context of longer sequences of text by retaining "memory" from previous tokens	- Slower to train, since it has to take longer to understand the context of a text sequence - Prone to the vanishing/exploding gradient problem
BERT	- "Plug and play", since it is already pre-trained on a large corpora - Can learn representations of words as vectors of high dimensionality - Obtain representations from both directions of a text sequence	- Since it is pre-trained on corpora that might not be suitable for a particular task, representations used in transfer learning might not be accurate

CHAPTER 3 System Methodology/Approach

3.1 Methodologies and General Work Procedures

The general methodology of the model in the proposed application is as follows. A data set consisting of plots and summaries of movies, along with the title of the movies is input into the model. Preprocessing is then performed on the features such as removing non-alphabetical characters and transforming all letters to lowercase. Then, stop words such as “and” and “or” are removed from the corpus. The vocabulary is then stemmed and lemmatized, meaning converted to their root forms. The dataset is then split into a training set and test set with an 80:20 ratio. The sets are also shuffled before splitting to prevent the model from accessing only the first during training.

Next, the target labels in the dataset, in this case, the movie titles, are encoded as integers. Then, a tokenizer is fitted on the plot text. This transforms the words in the text into a sequence of integers. The sequences are then padded with zeros to a fixed length, to ensure that each sentence has the same size. With this, the input is ready to be fed into the deep learning model, which goes through a series of layers for feature extraction.

After training, the model is ready to predict movie titles given an input sentence. The input sentence goes through the same preprocessing steps as above, and reaches the final dense layer with a softmax activation function, which will fire one of the possible neurons in it, corresponding to the label of the predicted movie title.

3.2 User Requirements

- Users are expected to input coherent, understandable English text.
- Users can only input plots/summaries from movies that exist in the data set.
- Users are able to identify the movie they are thinking of from the results returned.

3.3 System Performance Definition

The main measure of the performance of the proposed application is the accuracy in predicting movie titles from plot or summary descriptions. The accuracy of the predictions can be influenced by multiple factors, including the number of words that can uniquely identify each movie as well as the sample size of input words used during model training. If a movie has a generic plot with not much variation in words used, it would be harder to identify it. The sample size of words that can identify a certain movie also has to be reasonably large, otherwise, it would be difficult for the model to tell different movies apart.

The fairness in the sample size of the input must also be considered because it might affect the overall accuracy of the model as well. Since the plots, summaries, and synopses are scraped from the Internet, with a majority of the text being contributed by users, there is a possibility that movies with a larger fan base will have more text associated with them, such as more detailed summaries and elaborate explanations, as those movies are more popular. As such, the model might be more biased towards movies that have higher popularity. To improve the accuracy of the model, each movie's plot or summary should have roughly the same number of words.

3.4 Verification Plan

Overall, the intended outcome of the application is to successfully predict the title of the movie the user is thinking of based on plot/summary descriptions. The details of this verification plan will be further elaborated in Chapter 6. The verifications that need to be made are:

1. The user inputs a sentence describing the overall plot of the movie.

Table 3.1.5.1: Verification Plan

Method	Testing
Applicable requirements	Correctly identify movie title from a description of the overall plot of movie
Purpose/scope	To identify movie title from a description of the overall plot of movie
Items under test	Plot description/summary/synopses/screenplay
Precautions	The input sentence must not be too general
Special conditions/limitations	Vocabulary used in input is limited to the vocabulary used in dataset
Equipment/facilities	Laptop
Data recording	None
Acceptance criteria	Application correctly identifies movie title
Procedures	1. User inputs sentence 2. User observes returned results and sees if they contain intended movie
Troubleshooting	Go through more iterations by predicting different movies and using increasingly vague sentences
Post-test activities	None

CHAPTER 4 – System Design

4.1 System design flowchart

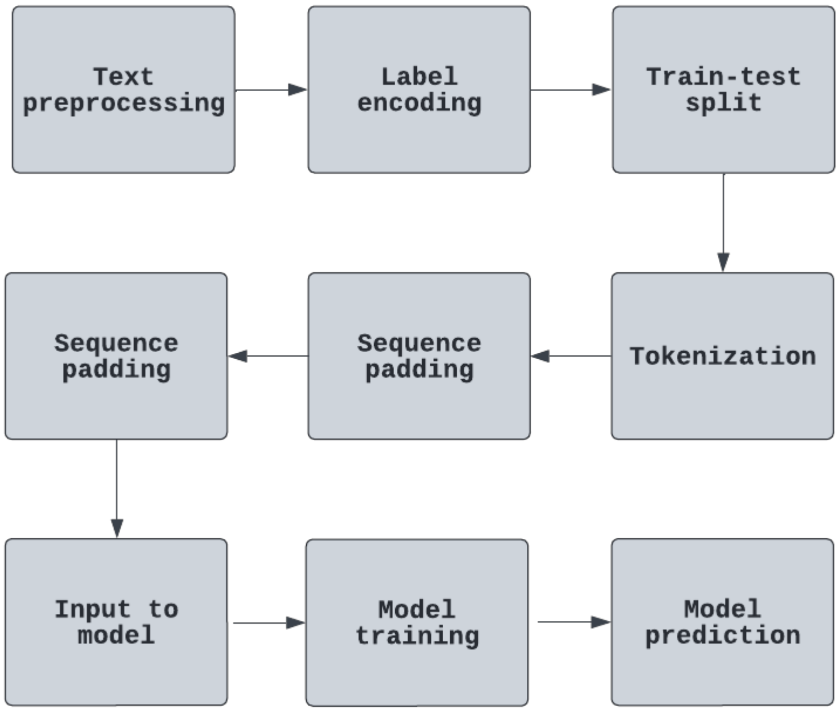


Figure 4.1.1 – System design flow chart

4.2. Deep learning pipeline

This section will outline the steps taken in the deep learning pipeline for the development of the model.

4.2.1 Data acquisition

This is technically the first step towards building the system for this project, but it is not directly a part of the system. Nevertheless, the steps taken to acquire the dataset will be gone through.

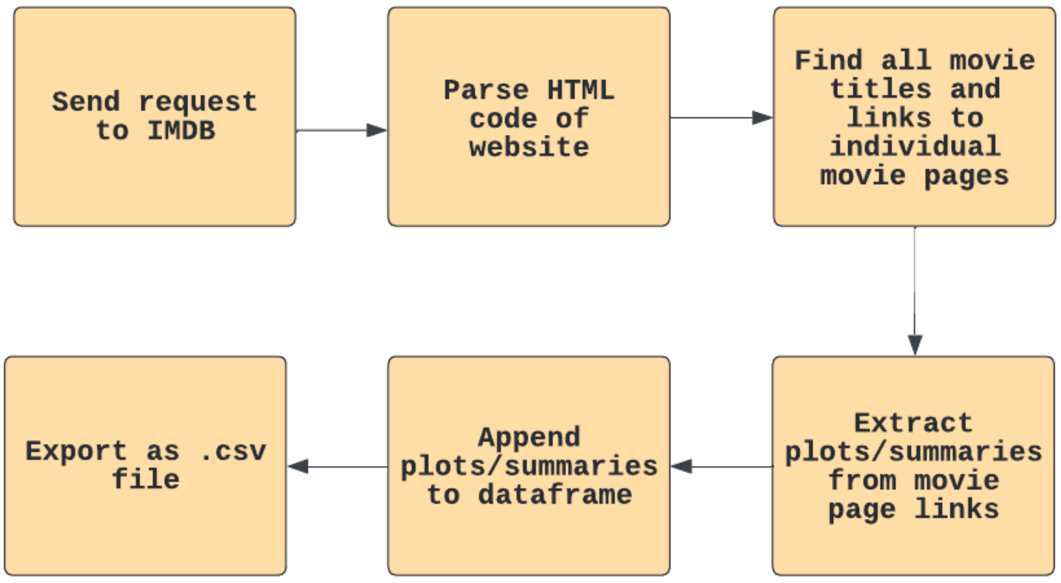

Send request to IMDB
Parse HTML code of website
Find all movie titles and links to individual movie pages
Extract plots/summaries from movie page links
Append plots/summaries to dataframe
Export as .csv file

Figure 4.2.1 – Process of scraping text data for dataset

With the help of Python libraries such as BeautifulSoup and requests, it is relatively simple to acquire the data needed for this project. Firstly, a GET request is sent to the IMDB website of top movies, because it contains the titles of those movies as well the links to each movie page (containing plots and summaries). By parsing the HTML code of the first website, the movie titles and links to those movie pages can be appended to a list. By iterating over the list, GET requests can be sent with a URL created by concatenating the main IMDB website URL header (http://www.imdb.com) and the URL of the individual movie pages ("/title/tt6710474/"), for example. Now that access to the movie page is achieved, HTML parsing can be done again to extract only the plots/summaries from the related section. Finally, the plots/summaries are appended to a Pandas dataframe, and exported as a .csv file.

4.2.2 Text preprocessing

In order to reduce redundant information within the corpus, several preprocessing steps are taken. Firstly, every other character that is not part of the alphabet is removed, such as numbers and symbols. All the words are also converted to lowercase. To reduce the number of unique words, stemming and lemmatization are applied to the text. Stemming literally “cuts” off the last few characters of words that are similar but in different tenses. For example, “eat” and “eating” are reduced to simply “eat”. However, “ate” is related to “eat”, but they do not share a common string, and are therefore unaccounted for in stemming. This is where lemmatization comes in. Lemmatization also reduces similar words to their root form, but taking into account the validity of the reduced word, ensuring that the final product actually exists in the dictionary. Finally, stop words are removed from the corpus, such as “a”, “and” or “the”. These words do not contribute much meaning towards the overall text and only increase the size and noise of the corpus.

4.2.3 Label encoding

Deep learning models can only comprehend numerical data, but the dataset consists solely of text. To overcome this, the target labels, in this case, the movie titles, are encoded as integers. This technique maps an integer to each unique movie title, but not for each appearance in the dataset.

4.2.4 Train-test split

The dataset is split into the training set and test set with a ratio of 80:20. The training set will be fed into the model, whereas the test set will be used for validation.

4.2.5 Tokenization

Tokenization breaks down long sentences into a list of individual words, commonly separated by whitespace. After tokenizing the input to our model, which are the plots/summaries, we end up with lists instead of strings. After the tokenization process, a Python dictionary of the vocabulary is generated, with its keys and values corresponding to the words and integers.

4.2.6 Text to sequence

Similar to the movie titles, the input to the model has to be somehow converted to numerical values. From the tokenization process mentioned previously, each word in the vocabulary now has a corresponding integer value. With this, the tokenized lists can be converted into lists of integers. Conversely, these lists of integers can be reversed back into words using the same dictionary.

4.2.7 Sequence padding

Deep learning models only accept batches of input of a fixed length. However, not all sentences in the dataset have the same length. To standardize the length of the input, each list can be padded with zeros up to the length of the longest sentence in the entire dataset. This ensures that no information is cut off from other sentences.

4.2.8 Model architecture

Now that the input is prepared in a suitable format, it is ready to be fed into the model. In this project, since numerous model architectures will be tested, a general diagram will be illustrated for the CNN and RNN architecture, whereas another diagram will be illustrated for the BERT architecture.

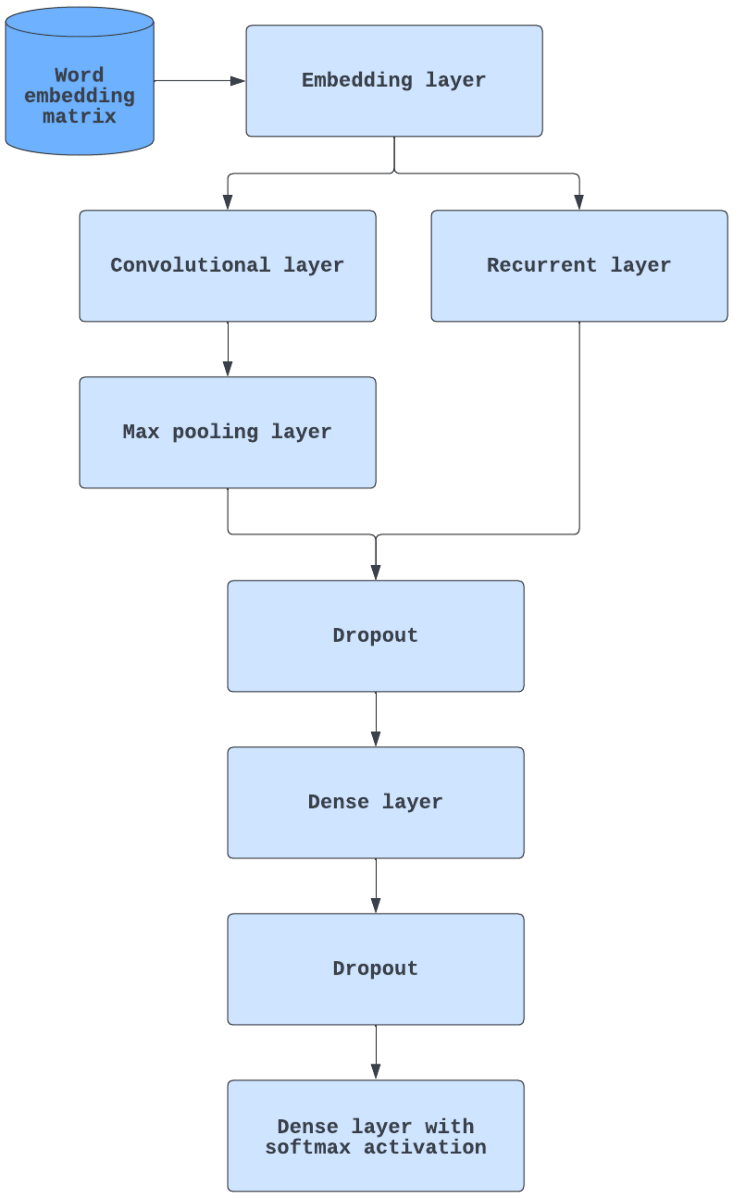


Figure 4.2.3 – General deep learning neural network model flowchart

For both models, the first layer is an embedding layer, with an input dimension of the vocabulary size. The output dimension instead will depend on the size of the next layer. The input length of the embedding layer is the maximum padded length of the input. The weights fed into the embedding layer are from the word embeddings matrix generated from Word2Vec or GloVe. These weights will be used to produce more accurate word embeddings in relation to the dataset.

In a CNN model, the next layer will be a 1D convolutional layer, with parameters such as number of filters, kernel size, stride and activation function, followed by a max pooling layer, which downsamples the learned features and keeps only the most important ones. On the other hand, in an RNN model, the next layer is a SimpleRNN layer, with the most important parameter being the number of RNN units. In both models, multiple instances of these layers can be stacked upon one another.

A practice applied in this project is to apply a dropout layer after every few layers, which randomly disables certain inputs from the previous layer. This prevents the model from learning from all of the features and reduces overfitting. It only has one parameter, which is a value from 0 to 1, denoting the percentage of values to drop from the previous layer.

The next layer added is just a regular dense layer, with a varying number of neurons. The number of neurons in the dense layer can be adjusted to a large number, which might capture more complex patterns in the text, but might take longer to train. The dense layers can be combined with dropout layers and repeatedly added many more times in the model.

The final layer is also a dense layer, but the number of neurons has to be the same as the number of movie titles in the dataset. This is because this is a classification layer, which outputs the probabilities of the input to the model correlating to each movie. The neuron with the largest value in its output is the model's prediction for the most likely movie the input is referencing.

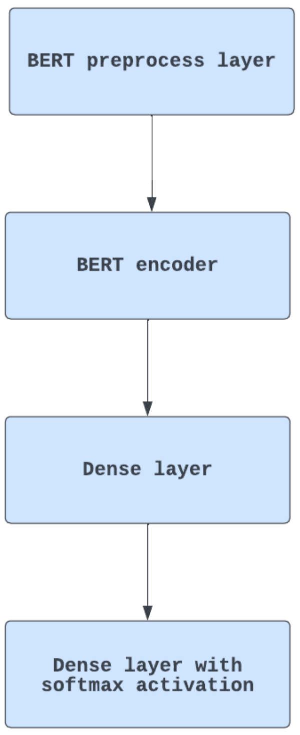


Figure 4.2.4 – BERT model flowchart

In contrast to regular neural network architectures, the BERT architecture is much more straightforward to implement, due to it already being pre-trained. The BERT preprocess layer and encoder can be downloaded from the official Tensorflow hub website and used in the model directly. Another notable feature of using transfer learning with BERT is that the text does not necessarily need to go through preprocessing steps as shown previously, as BERT can handle that by itself. Next, the preprocessed input goes through the encoder, where high level representations of each token are learned and extracted. Just for good measure, a dense layer is added after the encoder. Finally, the features go through a classification layer with softmax activation, similar to the regular models.

4.3 Model prediction

Although the CNN/RNN models differ from the BERT model in architecture, the prediction process is similar, due to both models being built with Keras. Using the built-in predict function from Keras, the model generates a list of probabilities for all movies in the dataset. The predictions are in the form of floating point values ranging from 0 to 1. After acquiring the indexes of the highest probabilities, the label encoder is used to look up the movie titles according to the integer labels.

CHAPTER 5 – System Implementation

5.1 Hardware setup

Table 5.1.1– Laptop specifications

Description	**Specifications**
Model	Acer Nitro 5 (AN515-42)
Processor	AMD Ryzen 5 2500U
Operating System	Windows 11
Graphic	AMD Radeon RX 560X
Memory	16GB DDR4 RAM
Storage	1TB SATA HDD, 240GB SSD

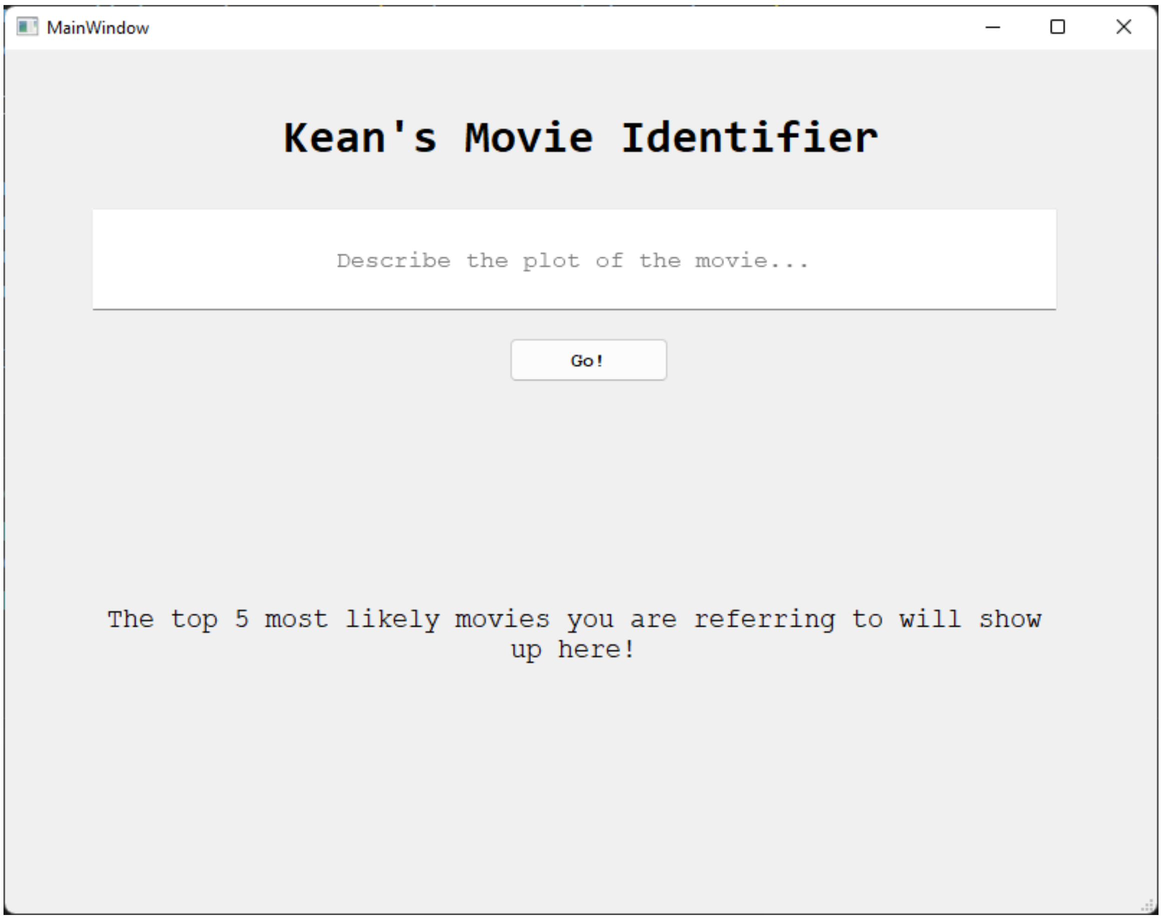
5.2 Software setup

1. Google Colab
2. Visual Studio Code

The majority of the coding work is done in Google Colab, due to its advantage in using Google's cloud GPU, making model training time faster. However, generating the application GUI as an .exe is impossible in Google Colab, hence the code has to be downloaded and run on a local IDE, such as Visual Studio Code.

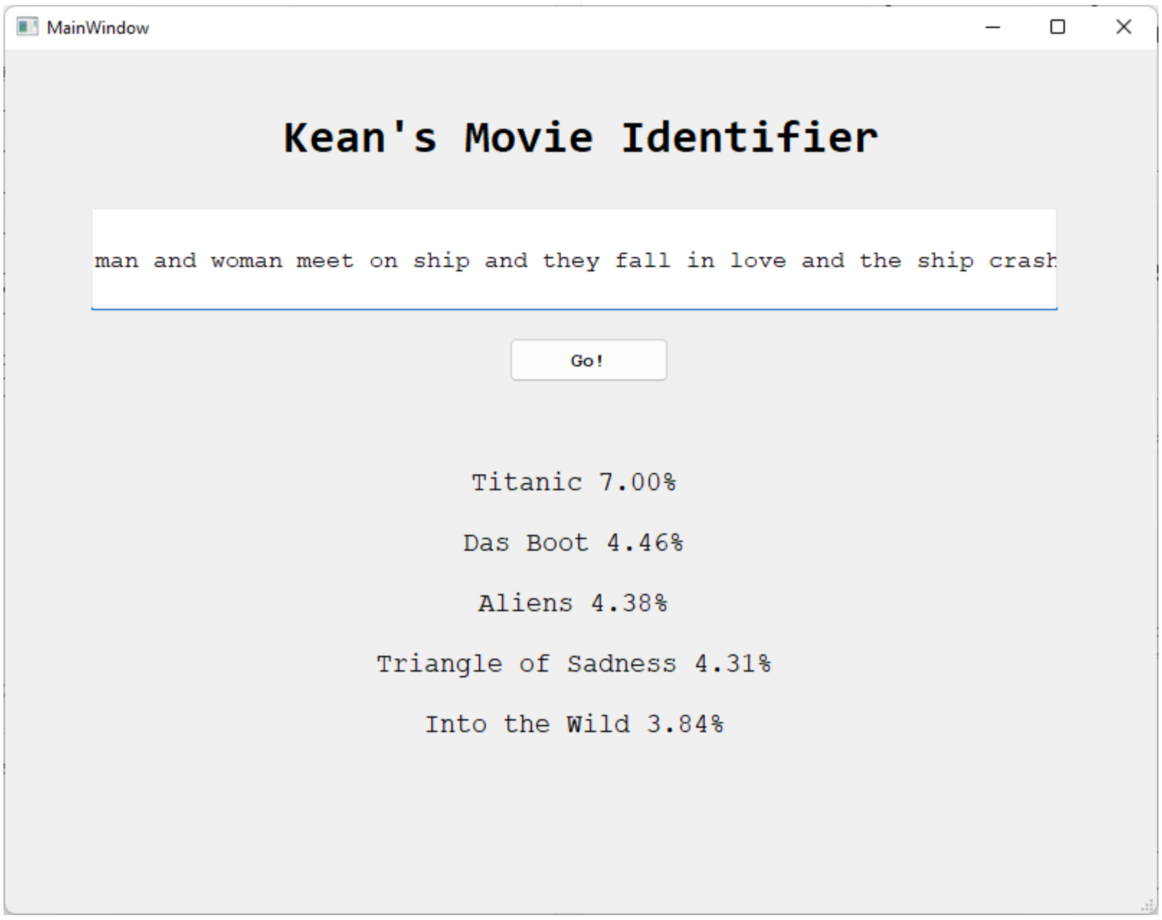
5.3 System Operation

The application has a simple interface, and requires only one input from the user, which is text related to the plot of a movie. After providing the input text, the top 5 predicted movie titles and their corresponding probabilities are displayed. The user can input text as many times as they wish.



The top 5 most likely movies you are referring to will show up here!

Figure 5.3.1 – Initial state of application



```
man and woman meet on ship and they fall in love and the ship crash

Titanic 7.00%
Das Boot 4.46%
Aliens 4.38%
Triangle of Sadness 4.31%
Into the Wild 3.84%
```

Figure 5.3.2 – Returned predictions based on input

5.4 Implementation Issues and Challenges

Naturally, with the development of a project of such a scale, issues and challenges are faced. In the case of this project, it is unfortunate that the computer used does not have sufficient specifications to train a model of such complexity. Therefore, Google Colab is used due to its cloud computing services, albeit with a downside of having limited time for each session.

Both the CNN and BERT models took around 30 minutes to train, which is still satisfactory. However, the RNN model took an unreasonably large amount of time to train, which is unrealistic in terms of this project. Therefore, the focus has been placed on the CNN and BERT models instead.

Lastly, the major challenge faced in the implementation of this project is the model's ability to predict movie titles based on descriptions of its plot correctly. This will be further elaborated on in the next chapter. In a nutshell, the prompt given has to be specific in order for the model to make an accurate guess.

5.5 Concluding remarks

Despite the shortcomings in the implementation of this project mentioned above, it was still an overall success, as a feasible application is developed and reasonably achieves its primary objective. Regrettably, the state of application development in deep learning still requires powerful hardware to alleviate poor training times, which is not something that is accessible to everyone. Ultimately, workarounds have to be discovered to bypass these issues.

CHAPTER 6 – System Evaluation and Discussion

6.1 System testing and performance metrics

The models proposed can be evaluated in two ways. The first method is the most common technique to evaluate any deep learning model, which is observing its training accuracy and validation accuracy during training. The second method is by actually performing inference with the model via different plot descriptions as stated in the verification plan proposed above.

Three models are to be evaluated, which are CNN-Word2Vec, CNN-GloVe, and BERT. The RNN model proposed cannot be trained due to the limitations mentioned earlier.

6.2 Test results

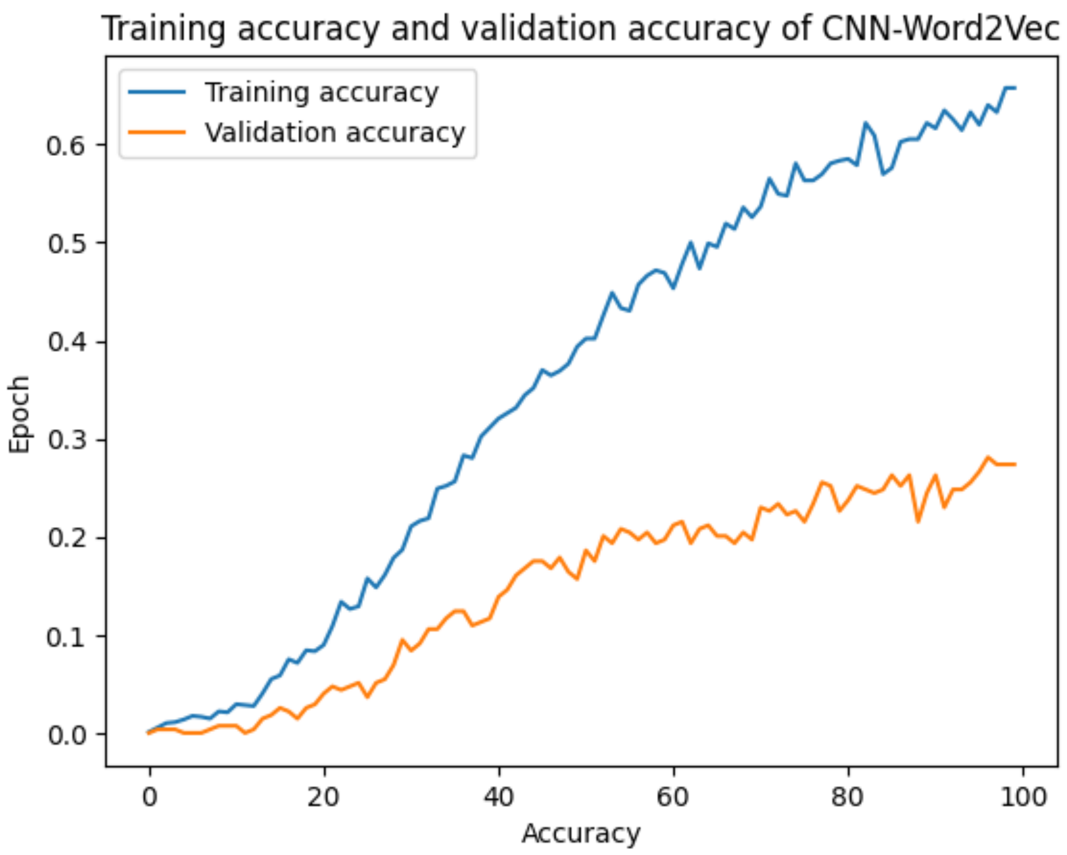


Figure 6.2.1 – Performance of the CNN-Word2Vec model

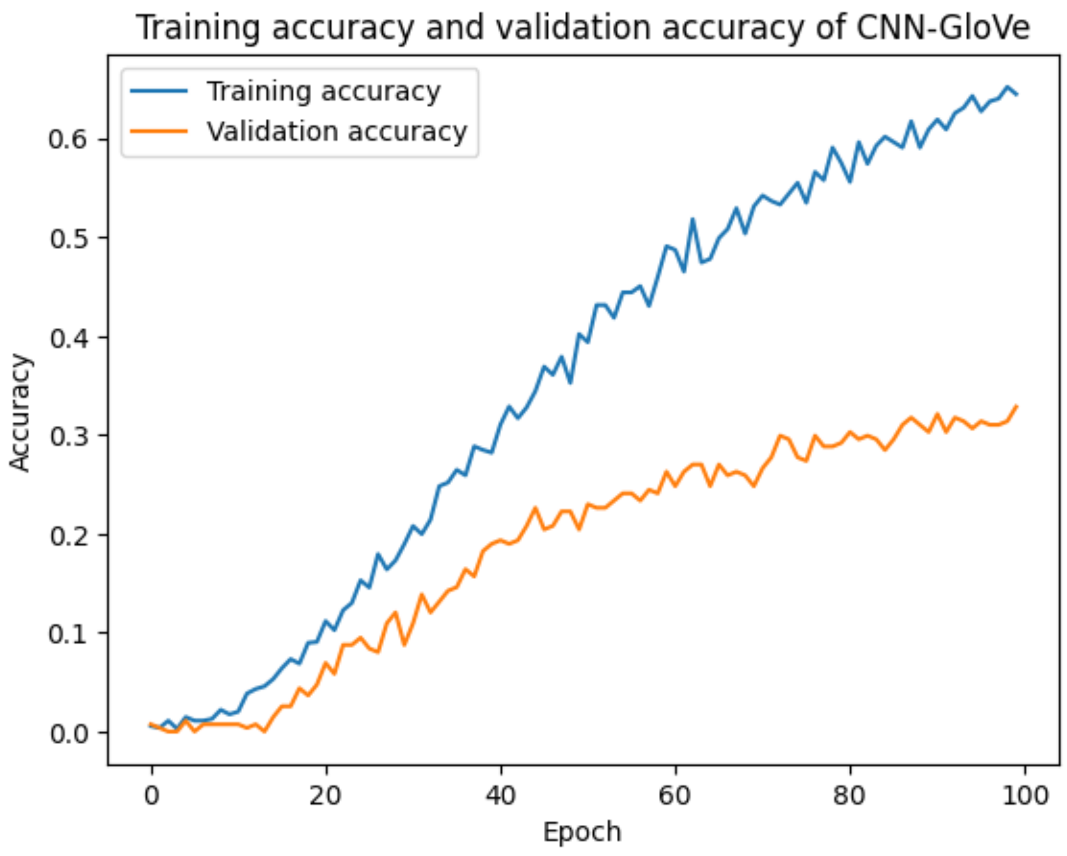


Figure 6.2.2 – Performance of the CNN-GloVe model

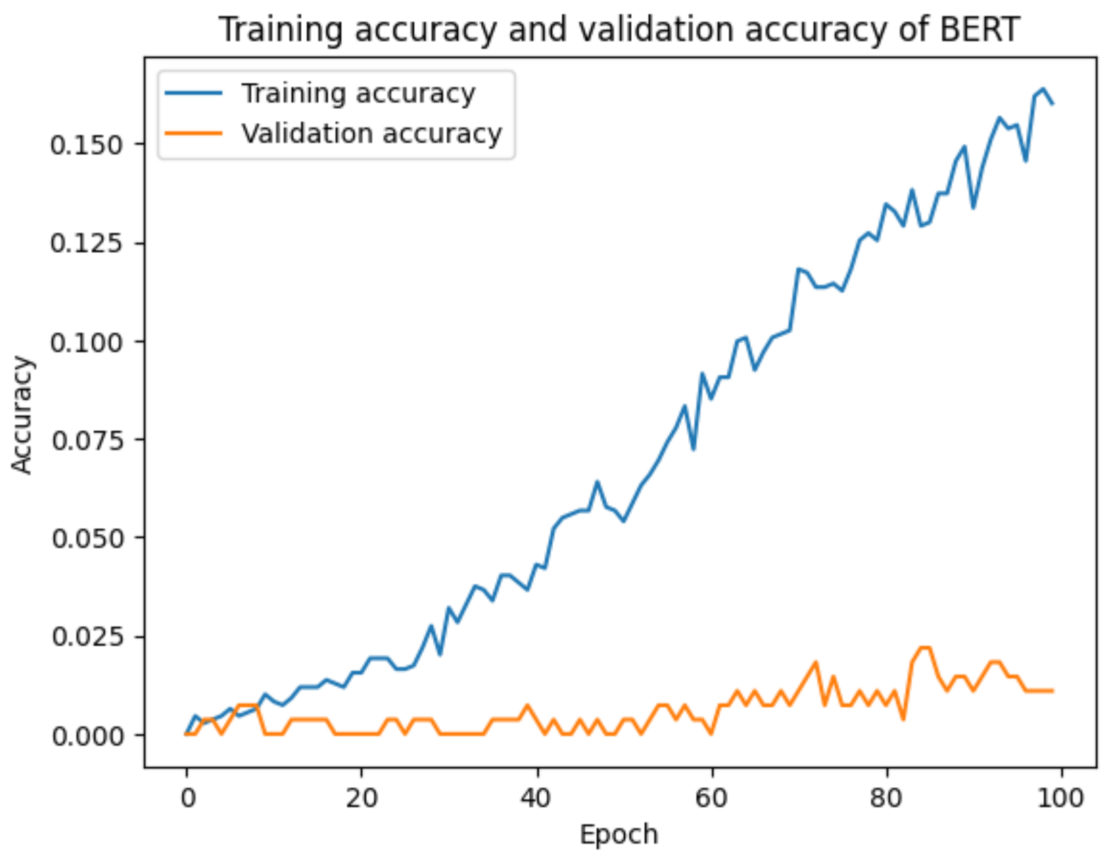


Figure 6.2.3 – Performance of the BERT model

Immediately, some surprising observations can be made based on the graphs above. The performance of the BERT model is abysmal compared to the two CNN models with different word embeddings above. As for comparisons between the two CNN models, nothing of significance can be made based on the values of both training accuracy and validation accuracy.

The next stage of evaluation will put these three models to the test by providing input sentences describing the plot of several movies and comparing their predictions.

For this evaluation, the intended movie is *Raiders of the Lost Ark (1981)* starring Harrison Ford as Indiana Jones. The first sentence is relatively straightforward, and omits multiple details from the plot of the movie.

"harrison ford plays an archaeologist and fights nazis for ancient treasure"

Table 6.2.1 – Simple plot prediction for Raiders of the Lost Ark

CNN-Word2Vec	["Howl's Moving Castle"] 12.32% ['Monty Python and the Holy Grail'] 4.04% ['Puss in Boots: The Last Wish'] 3.36% ['Demon Slayer: Kimetsu No Yaiba - To the Swordsmith Village'] 2.43% ['The Goonies'] 2.05%
CNN-GloVe	['Indiana Jones and the Temple of Doom'] 31.04% ['Spirited Away'] 5.42% ['The Goonies'] 5.37% ['WALL·E'] 4.47% ['Monty Python and the Holy Grail'] 3.52%
BERT	['Harry Potter and the Deathly Hallows: Part 2'] 66.67% ['Avengers: Infinity War'] 13.59% ['Avengers: Endgame'] 9.41% ['Star Wars'] 1.32% ['The Whale'] 0.94%

From the predictions above, although none of the models got it right, CNN-GloVe is the closest to identifying the intended movie. It recognized important keywords, such as Harrison Ford playing an archaeologist and ancient treasure, but it did not note the absence of Nazis as antagonists in its top prediction., "*Temple of Doom*". Nevertheless, this shows that at least one of the models trained has some vague notion of the plot of the Indiana Jones film series, as well as identifying important cast members.

The second sentence will provide more detail such as a memorable scene from the movie.

"*archaeologist travels the world to find some sort of chest and encounters many enemies along the way, one scene includes him pulling out a gun to shoot an enemy swordsman*"

Table 6.2.2 – Scene description inclusive plot prediction for Raiders of the Lost Ark

CNN-Word2Vec	['Bullet Train'] 11.98% ['The Incredibles'] 8.20% ['Amores Perros'] 5.06% ['Lock, Stock and Two Smoking Barrels'] 4.76% ['The Iron Giant'] 3.61%
CNN-GloVe	['Pirates of the Caribbean: The Curse of the Black Pearl'] 9.24% ['Alien'] 5.01% ['The Iron Giant'] 4.72% ['Top Gun: Maverick'] 3.77% ['1917'] 3.24%
BERT	['Star Wars'] 32.27% ['Oldboy'] 22.48% ['Avengers: Infinity War'] 10.45% ['The Flash'] 10.12% ['Avengers: Endgame'] 4.17%

Unfortunately and unintuitively, in this particular example, providing more details regarding the plot has apparently confused the model more than without it. It can be inferred from this result that the models, or at least CNN-GloVe (since it was somewhat close in its prediction for the previous sentence), rely on rare and unique keywords that can identify a particular movie well, such a "*nazi*" and "*treasure*" , rather than the overall comprehension of the plot of a movie.

6.3 Project challenges

Similar to the implementation challenges earlier, the primary reason affecting the performance of the model is the lack of a large enough dataset. As stated by the distributional hypothesis, words that occur together in some body of text tend to have a similar context. By using a relatively small dataset, the models struggle to form coherent connections between word vectors to associate keywords input by the user with the words that represent each plot in the dataset in a meaningful way.

Also, the lack of computing power and resources makes it challenging to train models with high complexity, and models that are deep enough to infer semantic meaning from a handful of words. As such, the models trained in this project only have a "surface level" understanding of the text in the dataset, such as memorizing certain keywords that uniquely appear for only a few movies.

6.4 Concluding remarks

It is difficult to say if the objectives set in this project are completely achieved, but it is safe to conclude that one of the models presented, which is CNN-GloVe, has demonstrated a basic level of understanding of the natural language text provided to it, as well as made reasonably close predictions. This is for the most part due to the usage of GloVe word embeddings that allow the model to learn word representations to accommodate the difference in the vocabulary used by users and the corpus used to train it.

CHAPTER 7 – Conclusion

This project set out to help lovers of cinema identify movies they have watched in the past but have forgotten their titles. With the proposed models in this project, it is hoped that this goal can be achieved, or at least make some contribution towards solving this problem.

To recap, this project utilized several natural language processing and deep learning techniques, such as word embeddings to construct connections and representations between words, and neural networks to extract and learn features and patterns within the text.

Though the models proposed might not have significant performance in terms of multiclass text classification, some key findings are indeed discovered in terms of how the model associates certain words with certain movies, which is through identifying words that are unique to a particular sample. This is of course not the most optimal way to learn the relationships between different words, but this is a small step towards achieving that goal.

To further improve the performance of the models proposed, the most impactful way to do so would be to acquire a larger dataset for the model to train on. This would not only allow the model to look at more samples to avoid overfitting, but also help it generalize well to a wider range of vocabulary as well.

May the findings from this project be of help to some people, be it enthusiasts of natural language processing and deep learning, or lovers of cinema.

REFERENCES

[1] J. Arguello, A. Ferguson, E. Fine, B. Mitra, H. Zamani, and F. Diaz, “Tip of the Tongue Known-Item Retrieval: A Case Study in Movie Identification,” *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*, 2021.

[2] M. Akser, “Cinema, life and other viruses: The future of filmmaking, film education and film studies in the age of covid-19 pandemic,” *CINEJ Cinema Journal*, vol. 8, no. 2, pp. 1–13, 2020.

[3] Elokence, “Akinator,” *en.akinator.com*. [Online]. Available: https://en.akinator.com. [Accessed: 17-Aug-2022].

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv.org*, 07-Sep-2013. [Online]. Available: https://arxiv.org/abs/1301.3781. [Accessed: 24-Apr-2023].

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *arXiv.org*, 16-Oct-2013. [Online]. Available: https://arxiv.org/abs/1310.4546. [Accessed: 24-Apr-2023].

[6] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors forWord Representation,” *Glove: Global vectors for word representation*, 2014. [Online]. Available: https://nlp.stanford.edu/projects/glove/. [Accessed: 25-Apr-2023].

[7] Y. Kim, “Convolutional neural networks for sentence classification,” *arXiv.org*, 03-Sep-2014. [Online]. Available: https://arxiv.org/abs/1408.5882. [Accessed: 25-Apr-2023].

[8] T. Young, D. Hazarika, S. Poria, and E. Cambria, “Recent trends in deep learning based natural language processing,” *arXiv.org*, 25-Nov-2018. [Online]. Available: https://arxiv.org/abs/1708.02709. [Accessed: 25-Apr-2023].

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” *arXiv.org*, 06-Dec-2017. [Online]. Available: https://arxiv.org/abs/1706.03762. [Accessed: 25-Apr-2023].

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *arXiv.org*, 24-May-2019. [Online]. Available: https://arxiv.org/abs/1810.04805. [Accessed: 25-Apr-2023].

APPENDIX

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 4**
Student Name & ID: Kean Soh Zhe Herng - 19ACB01352	
Supervisor: Dr. Jasmina Khaw Yen Min	
Project Title: An Application for Identifying Movies From Plot with Word Embeddings and Deep Learning	

1. WORK DONE

- Reviewed literature on possible word embedding techniques and neural network architectures that can be implemented

2. WORK TO BE DONE

- Data acquisition, scraping plot/summaries and movie titles to construct dataset

3. PROBLEMS ENCOUNTERED

- Imbalance of sample size for each movie scraped, since some movies are popular than others and therefore will have more associated text

4. SELF EVALUATION OF THE PROGRESS

- Progress is slightly impeded due to other coursework, but otherwise still manageable

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 6**
Student Name & ID: Kean Soh Zhe Herng - 19ACB01352	
Supervisor: Dr. Jasmina Khaw Yen Min	
Project Title: An Application for Identifying Movies From Plot with Word Embeddings and Deep Learning	

1. WORK DONE - Successfully trained CNN model on dataset
2. WORK TO BE DONE - Train RNN model on the same dataset and compare performances
3. PROBLEMS ENCOUNTERED - RNN model training too slow in comparison with CNN model
4. SELF EVALUATION OF THE PROGRESS - Still on track, considering there is only one more model left to experiment with, which is BERT.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.:** 8
Student Name & ID: Kean Soh Zhe Herng - 19ACB01352	
Supervisor: Dr. Jasmina Khaw Yen Min	
Project Title: An Application for Identifying Movies From Plot with Word Embeddings and Deep Learning	

1. WORK DONE - Trained and tested all three proposed model architectures
2. WORK TO BE DONE - Develop GUI and begin writing report
3. PROBLEMS ENCOUNTERED - The PyQt5 code cannot be run on Google Colab, therefore the GUI can only be viewed if the code is run with an IDE on local device, such as Visual Studio Code
4. SELF EVALUATION OF THE PROGRESS - Progress is satisfactory, but the recommended chapter structure according to the report guidelines for Project II is not necessarily the most suitable for this project. Will consult supervisor on this.

[illegible] [illegible]

Supervisor's signature   Student's signature

POSTER

UTAR UNIVERSITI TUNKU ABDUL RAHMAN

Faculty of Information and Communication Technology

An Application for Identifying Movies From Plot with Word Embeddings and Deep Learning

by Kean Soh Zhe Herng

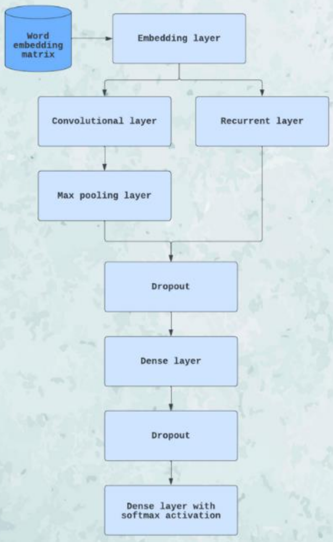
Introduction

This paper proposes an application for users to identify movies based on short descriptions of their plots. The model is built with natural language processing and deep learning techniques, such as word embeddings and neural networks.

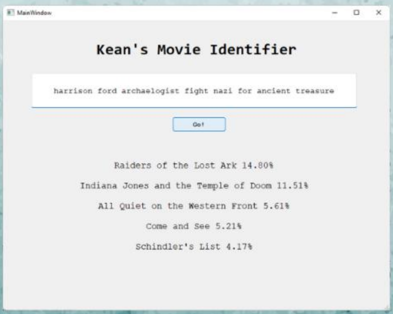
Objectives

1. To identify the top 5 titles of the movie described by the user, limited only to movies that exist in the dataset
2. To recommend other similar movies based on the returned predictions.

Model architecture



Results



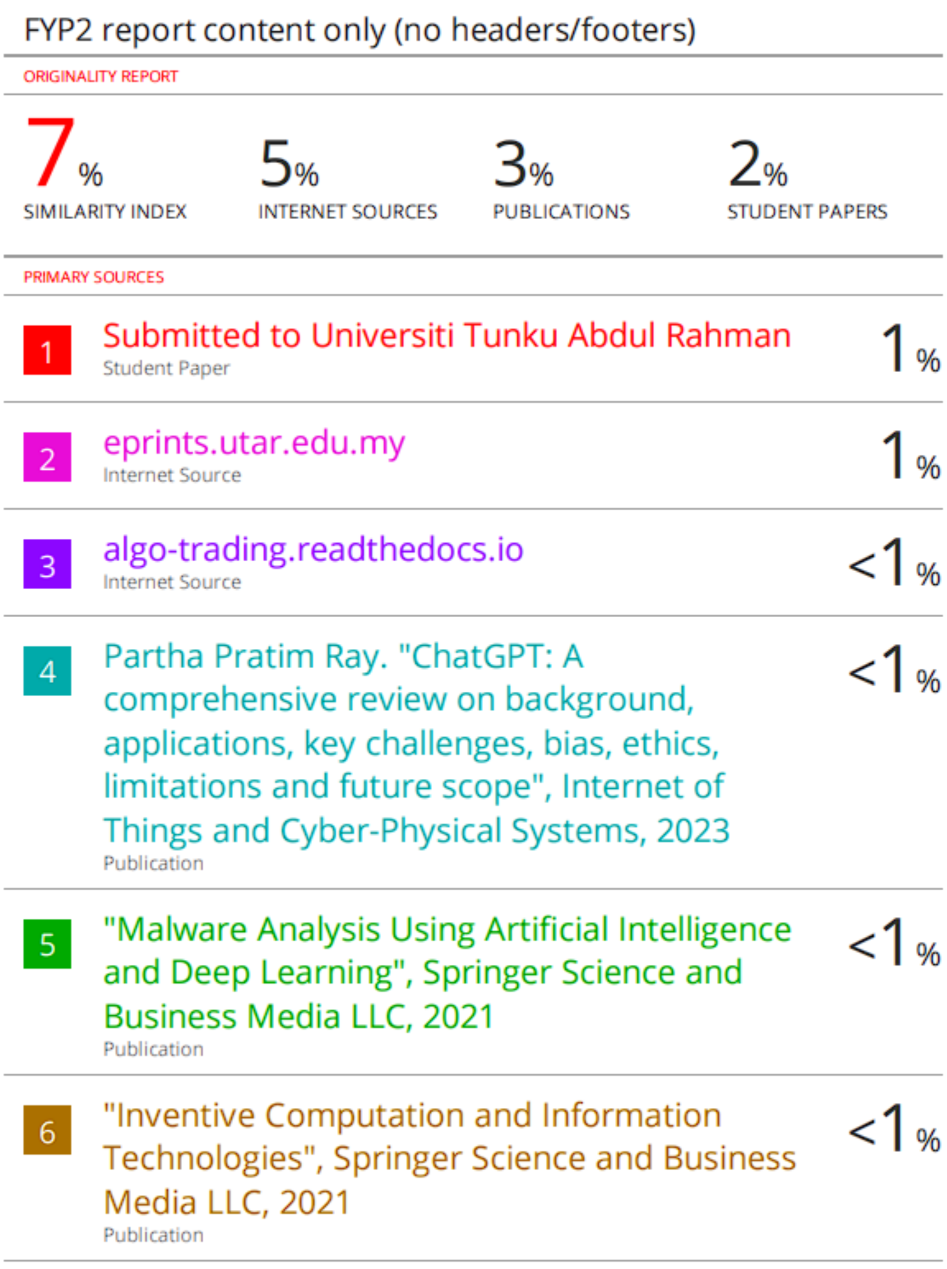
Kean's Movie Identifier

harrison ford archaelogist fight nazi for ancient treasure

```
Raiders of the Lost Ark 14.80%
Indiana Jones and the Temple of Doom 11.51%
All Quiet on the Western Front 5.61%
Come and See 5.21%
Schindler's List 4.17%
```

After input is provided by the user in the text box, the model will begin predicting the movie title based on features learned from the dataset. The top 5 predicted movie titles are displayed.

PLAGIARISM CHECK RESULT



FYP2 report content only (no headers/footers)

ORIGINALITY REPORT

7%	5%	3%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universiti Tunku Abdul Rahman Student Paper	1%
2	eprints.utar.edu.my Internet Source	1%
3	algo-trading.readthedocs.io Internet Source	<1%
4	Partha Pratim Ray. "ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope", Internet of Things and Cyber-Physical Systems, 2023 Publication	<1%
5	"Malware Analysis Using Artificial Intelligence and Deep Learning", Springer Science and Business Media LLC, 2021 Publication	<1%
6	"Inventive Computation and Information Technologies", Springer Science and Business Media LLC, 2021 Publication	<1%

7	iopscience.iop.org Internet Source	<1 %
8	arxiv.org Internet Source	<1 %
9	uia.brage.unit.no Internet Source	<1 %
10	web.archive.org Internet Source	<1 %
11	www.hindawi.com Internet Source	<1 %
12	Chung-Chian Hsu, Wei-Cyun Tsao, Arthur Chang, Chuan-Yu Chang. "Analyzing mixed-type data by using word embedding for handling categorical features", Intelligent Data Analysis, 2021 Publication	<1 %
13	Submitted to University of Stellenbosch, South Africa Student Paper	<1 %
14	Submitted to Hong Kong Baptist University Student Paper	<1 %
15	www.scotsman.com Internet Source	<1 %
16	Submitted to Kings College Student Paper	<1 %

17	Ling Bai, Tianzhong Zhang. "Complex deformation pattern of the Pamir–Hindu Kush region inferred from multi-scale double-difference earthquake relocations", Tectonophysics, 2015 Publication	<1 %
18	biblio.ugent.be Internet Source	<1 %
19	dr.ntu.edu.sg Internet Source	<1 %
20	hdl.handle.net Internet Source	<1 %
21	ijict.itrc.ac.ir Internet Source	<1 %
22	scholarworks.sjsu.edu Internet Source	<1 %
23	www.moviejungle.com Internet Source	<1 %
24	Lecture Notes in Computer Science, 2015. Publication	<1 %
25	Marek Deja, Isto Huvila, Gunilla Widén, Farhan Ahmad. "Seeking innovation: The research protocol for SMEs' networking", Heliyon, 2023 Publication	<1 %
26	deepai.org Internet Source	

		<1 %
27	hal.archives-ouvertes.fr Internet Source	<1 %
28	jwcn-eurasipjournals.springeropen.com Internet Source	<1 %
29	par.nsf.gov Internet Source	<1 %
30	users.encs.concordia.ca Internet Source	<1 %
31	publikasi.dinus.ac.id Internet Source	<1 %
32	pure.uva.nl Internet Source	<1 %
33	www.researchgate.net Internet Source	<1 %

Form Number: FM-IAD-005	Rev No.: 0	Effective Date:	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Kean Soh Zhe Herng
ID Number(s)	19ACB01352
Programme / Course	Bachelor of Computer Science (Honours)
Title of Final Year Project	An Application for Identifying Movies From Plot with Word Embeddings and Deep Learning

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)**
Overall similarity index: 7% **Similarity by source** Internet Sources: 5% Publications: 3% Student Papers: 2%	
Number of individual sources listed of more than 3% similarity: 0	

Parameters of originality required, and limits approved by UTAR are as Follows:
(i) Overall similarity index is 20% and below, and
(ii) Matching of individual sources listed must be less than 3% each, and
(iii) Matching texts in continuous block must not exceed 8 words
Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.

Note: Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

[illegible]

Signature of Supervisor	Signature of Co-Supervisor
Name: Jasmina Khaw Yen Min	Name:
Date: 28/04/2023	Date:

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student ID	19ACB01352
Student Name	KEAN SOH ZHE HERNG
Supervisor Name	DR. JASMINA KHAW YEN MIN

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Plastic Cover (for hardcopy)
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
	List of Symbols (if applicable)
	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

[illegible]

(Signature of Student)
Date: 27 APRIL 2023