



Character Profiling in Literature through Natural Language Processing

Subin Jung (190398149)
May 2022

BSc Computer Science
Supervisor: Elly Liang

Word Count: 11233

Abstract

Artificial Intelligence, and especially natural language processing, promises to provide solutions in information extraction from text. Natural language processing can even enable character comprehension. While there have been existing research about character comprehension from various resources, this document reviews on the character profiling through literature. Taking a text classification approach, I propose a new design of literature character profiling system, which can guess speaker's name by a random given sentence. As profiling will be executed by training the dataset of speaker and dialogue, I have collected datasets from raw text by building the parser and applying natural language processing techniques and predict the speaker with five different classifiers, and have a reasonable degree of success. Finally, I consider how I could modify my character profiling model to increase accuracy and propose potential future directions for the project.

Declaration

“I declare that this dissertation represents my own work except where otherwise stated.”

Acknowledgements

I would like to thank my supervisor Dr Elly Liang for her invaluable support and guidance over this project.

Table of Contents

1. Introduction.....	7
1.1 Paper structure.....	7
1.2 Terminology.....	7
1.3 Motivation.....	8
1.4 Aim and Objectives	9
1.5 Changes.....	10
2. Background.....	10
2.1 Artificial Intelligence	11
2.2 Machine Learning	11
2.3 Text Classification	11
2.4 Natural Language Processing	12
2.4.1 Tokenisation	12
2.4.2 Part-of-Speech tagging.....	13
2.4.3 Dependency Parsing	14
2.4.4 Term Frequency-Inverse Document Frequency	15
2.5 Classification Model	15
2.5.1 Random Forest Classifier	15
2.5.2 Naïve Bayes Classifier	16
2.5.2.1 Gaussian Naïve Bayes Classifier.....	16
2.5.2.2 Multinomial Naïve Bayes Classifier.....	17
2.5.2.3 Bernoulli Naïve Bayes Classifier	17
2.5.2.4 Complement Naïve Bayes Classifier	17
2.6 Related Work.....	18
3. Design.....	19
3.1 Plan	19
3.2 Tool	19
3.2.1 Python.....	19
3.2.2 Google Colab.....	20
3.2.3 Scikit-learn	20
3.2.4 SpaCy.....	20
4. Implementation	20
4.1 Data Collection	21
4.1.1 Dialogue & Context Extraction	21
4.1.2 Natural Language Processing	22
4.1.3 Dialogue Tag Rule.....	23
4.2 Coreference Resolution	25
4.3 Pre-processing	27
4.3.1 TF-IDF Vectorization	27
4.4 Hyper-parameter tuning	28
4.4.1 GridSearchCV	28
4.5 Training	29
4.5.1 Random Forest Classifier	29
4.5.2 Gaussian Naïve Bayes Classifier	30
4.5.3 Bernoulli Naïve Bayes Classifier.....	30
4.5.4 Multinomial Naïve Bayes Classifier	30

4.5.5 Complement Naïve Bayes Classifier	30
4.6 User Interface	30
4.6.1 Ipywidget	30
5. Result and Evaluation	31
5.1 Dataset	32
5.2 Evaluation.....	32
5.2.1 Coreference Resolution.....	32
5.2.2 Hyper-parameter tuning	34
5.2.3 Confusion Matrix.....	36
5.2.4 Performance Metrics.....	39
5.3 User Interface	40
6. Conclusion	41
6.1 Overall Summary	42
6.1.1 Assessment of Aim and Objectives	42
6.1.2 Reflection	43
6.2 Future Work.....	43
References	45
Appendices	48
Appendix A.....	48
Appendix B.....	49
Appendix C.....	50
Appendix D.....	51
Appendix E.....	52
Appendix F	53

Table of Figures

FIG. 1. PROCEDURE OF NLP	12
FIG. 2. PROCESS OF TOKENISATION	13
FIG. 3. POS TAGGED TOKENS	13
FIG. 4. DEPENDENCY PARSING ON POS TAGGED TOKENS	14
FIG. 5. VISUALISATION OF RANDOM FOREST CLASSIFIER.....	16
FIG. 6. AGILE METHODOLOGY	19
FIG. 7. ENTIRE PROCESS OF THE PROJECT.....	21
FIG. 8. DATASET OF DIALOGUE AND CONTEXT	21
FIG. 9. CODE OF GET_CONVERSATIONS()	22
FIG. 10. CASE OF DIALOGUE TAGGING 1.....	24
FIG. 11. CODE OF EXTRACTING SPEAKER WITH NLP.....	24
FIG. 12. CASE OF DIALOGUE TAGGING 2.....	24
FIG. 13. CASE OF DIALOGUE TAGGING 3.....	25
FIG. 14. DATASET OF DIALOGUE, CONTEXT AND SPEAKER.....	25
FIG. 15. SCORE OF COREFERENCE RESOLUTION	26
FIG. 16. FINAL DATASET	26
FIG. 17. TF-IDF SCORES OF JOHN WATSON'S CORPUS.....	27
FIG. 18. WORD WEIGHT OF JOHN WATSON'S CORPUS	28
FIG. 19. HYPER-PARAMETER TUNING WITH GRIDSEARCHCV ON RANDOM FOREST CLASSIFIER	29
FIG. 20. BEST PARAMETERS SET ON RANDOM FOREST CLASSIFIER WITH GRIDSEARCHCV	29
FIG. 21. UI FOR GUESSING CHARACTER'S NAME.....	31
FIG. 22. AMOUNT OF DATA BEFORE SMOTE. (A) DATASET A. (B) DATASET B	32
FIG. 23. COMPARISON 'WITH AND WITHOUT COREFERENCE RESOLUTION' ON DATASET A.....	33
FIG. 24. COMPARISON 'WITH AND WITHOUT COREFERENCE RESOLUTION' ON DATASET B	33
FIG. 25. COMPARISON BEFORE AND AFTER HYPER-PARAMETER TUNING ON DATASET A.....	34
FIG. 26. COMPARISON BEFORE AND AFTER HYPER-PARAMETER TUNING ON DATASET B	34
FIG. 27. F1-SCORE COMPARISON BETWEEN DATASET A AND DATASET B.....	36
FIG. 28. CROSS-VALIDATION SCORE COMPARISON BETWEEN DATASET A AND DATASET B	36
FIG. 29. CONFUSION MATRIX OF DATASET A. (A) RF. (B) GNB. (C) BNB. (D) MNB. (E) CNB.	37
FIG. 30. CONFUSION MATRIX OF DATASET B. (A) RF. (B) GNB. (C) BNB. (D) MNB. (E) CNB.	38
FIG. 31. COMPARISON OF DATASET A	39
FIG. 32. COMPARISON OF DATASET B	40
FIG. 33. GUESSING SPEAKER'S NAME BY ENTERED SENTENCE.....	40
FIG. 34. CONFUSION MATRIX OF P0 AND P1	41
FIG. 35. CONFUSION MATRIX OF P3 AND P4	41

Table of tables

TABLE I.....	35
TABLE II.....	35

1. Introduction

This chapter will discuss the project's content by outlining the purpose as well as its aim and objectives. It begins with providing the paper structure to summarise each chapter and terminology as there will be many abbreviations. Additionally, as the project proposal was submitted, it will explain major changes.

1.1 Paper structure

The paper is structured as follows.

Chapter 1. Introduction presents the paper structure, terminology, motivation, aim and objectives, and changes.

Chapter 2. Background covers the previous research of the project. This chapter starts by introducing the background of AI and machine learning and provides an understanding of natural language processing and five different classification model and the related work of the project.

Chapter 3. Design focuses on the plan strategy and the detailed account of tools used for project's architecture.

Chapter 4. Implementation covers the architecture of the project and technology used and how. Also, it explains the strengths and limitations of chosen techniques. It contains the data collection, coreference resolution, pre-processing, hyper-parameter tuning, training and user interface.

Chapter 5. Result and Evaluation discuss the summary of the project evaluation and the comparison between every result made. It includes dataset, evaluation and user interface.

Chapter 6. Conclusion is the assessment of how the project's aim and objectives are accomplished. Additionally, it gives suggestions on any work that could be possibly done and proposes the future work.

1.2 Terminology

AI – Artificial intelligence is a branch of computer science designed to aid machines in the ability to appear intelligent or the ability to replicate intelligent human behaviour.

NLP – Natural Language Processing enables computers in comprehending human's natural language, especially dealing with ambiguities of vocabulary including homonyms, metaphors, and sarcasm.

Corpus – Corpus is a large collection of writings or recorded remarks used for linguistic analysis. In this project, it will simply refer to our ML dataset

Token – Tokens can be words, but can also be punctuation, or sub-units of words. It is common to speak of 'tokens' but to mean 'words'

POS tag – Part-of-Speech tagging is the process of categorising words in a sentence, in accordance with a specific part of speech, based on the word's definition and context.

TF-IDF – Term Frequency-Inverse Document Frequency is one of the numerical statistic methods which measures the importance of a word in a collection or corpus of documents.

RF – Random Forest Classifier is an estimator that employs averaging to increase predicted accuracy and control over-fitting by fitting a number of decision tree classifiers on various sub-samples of the dataset.

GNB – Gaussian Naïve Bayes is a machine learning algorithm based on the Bayes theorem and it is particularly employed when the features have continuous values.

BNB – Bernoulli Naïve Bayes is one of the Naïve Bayes algorithms used for machine learning. Especially, it is used for binary distribution classification.

MNB – Multinomial Naive Bayes is a machine learning algorithm using the Bayesian learning approach and it is suitable for text classification which has discrete features for each class.

CNB – Complement Naïve Bayes is one of the Naïve Bayes algorithms for machine learning and especially it is based on the Multinomial algorithm to correct its severe assumptions.

1.3 Motivation

In the last two decades, the development of highly successful machine learning (ML) technologies has dramatically evolved computational linguistics, known as the natural language processing (NLP) [1]. Natural language processing has been studied in various fields such as sentiment analysis [2] and character comprehension [3]. Among them, this paper especially concentrates on character profiling.

Character comprehension often happens to us unconsciously during reading. To be specific, when readers interpret the story, they not only create mental models of the situation and characters described but also perceive causal relationships between characters' behaviours and their intentions [4].

The main idea of developing a character profiling system is to give computer the language comprehension skills that are comparable to those of humans. For example, enabling to identify a character's traits means allowing to foresee how the character will react in a new situation. By adapting these abilities to computers, computers will be able to generate new vocabulary for certain characters by anticipating what they could say, which can lead to improving the machine's language system in the conversational AI [5].

Furthermore, although there has been recent research on character comprehension with the tv show [3] or social networking service (SNS) [6], character profiling with literature is unique as there is still no research done with this source in English. The reason why literature is chosen as the data source is that characters in literature are created by a human so it possesses the feature of real humans, which readers can sympathise with through their stories and emotions, even though they are not real.

Thus, this project's aim is to develop an artificial intelligence (AI) model that profiles characters in the literature by analysing their dialogues and applying different approaches to improve its performance. Additionally, the model can be evaluated by guessing a character's name from a given dialogue that is entered on the user interface (UI).

1.4 Aim and Objectives

The aim of this project is:

“Developing an AI model that profiles characters in the literature by analysing their dialogues and applying different approaches to improve its performance”

This will be achieved by completing 4 objectives as follows:

- **Collect a set of datasets from raw text using Natural Language Processing (NLP) techniques.**

This project focuses on analysing the dialogue from the literature. However, there is no previous example data collected. Thus, it is essential to collect the dataset of dialogues and speakers from the raw document. NLP will help to extract datasets by tokenising, POS tagging and dependency parsing on the text documents. As the literature is written in English, the dialogue rule of English literature will be adapted when building a text parser function.

- **Develop the model that can profile different kinds of literature so that results can be compared and analysed**

Profiling AI model will be tested with two different kinds of literature, Arthur Conan Doyle, *A Study in Scarlet* (1887) and Charlotte Brontë, *Jane Eyre* (1847). To analyse the prediction result, it is essential that the model can apply different documents so that the results can be compared and lead to feedback for enhancement.

- **Implement several popular classifiers on text data.**

There are lots of different machine learning classifiers, but it is essential to identify the classifiers which prove their accuracy with text type data. Also, it is important to compare and analyse the results between different classifiers so that the final model's accuracy can be increased.

- **Identify the best parameters for the model by implementing Hyper-parameter tuning**

Even if several classifiers are implemented, it is hard to say which one is the best as the performance might differ by their parameter values. Thus, applying Hyper-parameter tuning can find the best parameters for each classifier so that result comparison can be reliable. It is also sufficient to identify the appropriate tool for Hyper-parameter tuning.

- **Evaluating user interface (UI) based on the web application**

As the model will guess a character's name from a given dialogue, building a user interface (UI) for this task can let the viewer to approach it easily. When a user inputs a random sentence, the model will guess the character's name who likely said that the most. If the model profiles the characters successfully, then it will give the correct name. It is important to identify the proper tool for building UI.

1.5 Changes

As the project began, the detailed goals and motivation for achieving the aim have been changed. Whilst the initial aim is kept, the motivation and objectives are enhanced to focus on accomplishing the main goal of the project.

Previously, the goals are focused on obtaining knowledge and studying the research but they are shifted to concentrate on the methodology to reflect the aim of the project. Specifically, some of the objectives are mainly discussing what to investigate and how to refine the knowledge about technology.

However, as the main hypothesis of the project centred on the developing AI model and applying the different approaches to improve its performance, the objectives are replaced to focus on the development and analysis more.

Therefore, the developing process head to creating an AI model that has been improved during the experiment by adapting various techniques and approaches.

2. Background

This chapter will introduce the background knowledge, the main technologies that are used as approaches for the project as well as the review of the related works. It will start by introducing the background of AI, machine learning and text classification, then clarify the two main technologies, Natural Language Processing and Classification model. After that, it will carry out the similar research of this project.

Research Strategy

The research was carried out using both formal and informal sources. Mainly, formal sources are used to support this paper by referencing proven publication papers. Still, informal sources are used to assist the technologies and most of them are official websites of methods.

Proven publication research

- Google scholar and Newcastle University Library were used to find the research papers and books for supporting the background of AI, machine learning, natural language processing (NLP) techniques and classification models

Informal research

- Several reliable websites have been used such as IBM, GitLab, and the official site of SpaCy and Scikit-learn to support the principle of methodologies.

2.1 Artificial Intelligence

The root of AI can be traced back to the 1950s, Alan Turing, an English mathematician, invented the code-breaking machine and he released his article "Computing Machinery and Intelligence" in which he discussed how to build intelligent machines and how to test them. Then, the term AI was created in 1956 by Marvin Minsky and John McCarthy by organising the Dartmouth Summer Research Project on Artificial Intelligence (DSRPAI) at Dartmouth College in New Hampshire [7].

Nowadays, AI encompasses a wide range of computational techniques and processes aimed at boosting machines' capacity to perform tasks that require intelligence, such as pattern recognition, computer vision, and language processing [8]. With such a broad definition and the rapid advancement of technology, artificial intelligence is evolving over time. In addition, the terms 'machine learning' and 'deep learning' are often used interchangeably in most current references to AI, in fact, AI is a large category that includes machines and again deep learning is involved in the machine learning category [9].

2.2 Machine Learning

Machine Learning (ML), often known as artificial intelligence, refers to the intelligence that computers possess. It was a part of AI's evolution until the late 1970s. AI and machine learning are not new algorithms, AI has been around since the 1950s. One of the first machine learning programmes was created by IBM researcher Arthur Lee Samuels, who created a self-learning checkers programme. He even came up with the term "machine learning." In 1959, the machine learning approach was released by him as a report in the IBM Journal of Research and Development [10].

ML algorithm is a computing process that uses input data to accomplish a goal without being explicitly programmed to return the outputs. Naqa and Murphy [11] said, *"These algorithms are in a sense "soft coded" in that they automatically alter or adapt their architecture through repetition (i.e., experience) so that they become better and better at achieving the desired task."* The process, that they mentioned, is what we called training, which involves providing samples of input data together with intended consequences. After that, the algorithm then optimises itself so that it can not only provide the desired result when given the training inputs but also generalise to create the desired result when given new, previously unknown data. This process of training is the learning of machine learning.

Additionally, machine learning is utilised widely in several departments. From social media to intricate financial applications, machine learning is making its way into every facet of computing. Machine learning can be used to improve customer experience, better handle and forecast results from complex data, and even change how firms run. The ability to correlate data to find patterns and anomalies can aid in the prediction of outcomes and the improvement of processes [10].

2.3 Text Classification

To understand text classification, it is necessary to comprehend the extent of textual data and the meaning of classification. Specifically, Textual data can be anything from a phrase, a sentence, or a whole document containing paragraphs of text, and it can come from corpora, blogs, anywhere on the internet, or even an enterprise data warehouse. Text classification is sometimes known as document classification because the term *document* encompasses all types of textual material. Whilst the term *document* can refer to any concrete representation

of thoughts or events, such as writing, recorded speech, drawings, or presentations, text classification uses the term *document* to refer to textual data such as sentences or paragraphs in the language [12].

Text classification is another term for text categorisation. For two reasons, the word "classification" is used distinctly. The first reason is that it represents the same concept as text categorisation, which is used to organise papers. The second reason is to show that we are classifying or categorising text using classification or supervised machine learning. Text classification can be accomplished in a variety of ways, but this paper strongly emphasises on employing a supervised classification approach. The classification process is not limited to text and is utilised regularly in various fields such as research, healthcare, weather, and technology [12].

2.4 Natural Language Processing

Natural Language Processing (NLP) relates to a part of artificial intelligence (AI) and is providing the ability to computers to understand human language such as text data. It is the combination of statistical computational linguistics – such as rule-based modelling of human languages – and machine learning/deep learning models. In consequence, it helps the computer to acknowledge homonyms, metaphors or sarcasm despite the ambiguities of human language, by processing natural language in the form of text. As such, NLP is the technology that can let the computers complete the writer's intent and sentiment [13].

[15, Fig. 1.] illustrates the general procedure of NLP. Once the dataset is inputted, tokenisation is performed for splitting the sentence into tokens by tokenizer, and the tagger implements part-of-speech tagging to represent the attribute of each token. Then, the parser finds the dependency parsing between each token.

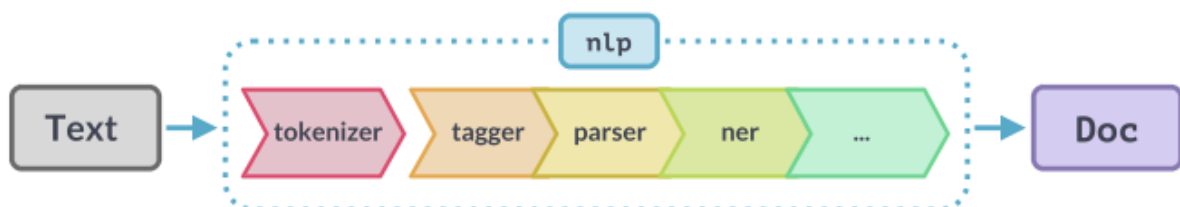


Fig. 1. Procedure of NLP

NLP has several state-of-the-art libraries such as Natural Language Toolkit (NLTK), SpaCy, Stanford's CoreNLP and Google's SyntaxNet. Omran and Treude [14] proposed that choosing the right NLP library can differ by the task and source. The detail of the tool chosen for this project will describe in 4.1.2 *Natural Language Processing*.

2.4.1 Tokenisation

Natural Language Processing (NLP) is the key technique to extract information from text data, and tokenisation is the first step of NLP. As the text data is segmented into smaller units named tokens, the separation process is called tokenisation.

Tokenisation has a huge difference from the string *split()* method in python. For example, while the *split()* method can only generate the list of tokens, tokenisation produces not only

the list of tokens but also the document of the token’s grammatical information, such as the type of the words and the relationships among the entities. Also, when separating punctuation from the sentence, whereas *split(“.”)* method not only separates the punctuation mark at the end of a sentence but also divides the word ‘U.K.’ into [‘U’, ‘.’, ‘K’, ‘.’], tokenisation recognises ‘U.K.’ as a single token [15].



2.4.2 Part-of-Speech tagging

This is a sentence.

DET VERB DET NOUN

However, the process of POS tag is not simple. Even though POS tag is performed, there could be a token which may not be tagged due to ambiguity. Thus, it is essential to use a

sequence model such as Hidden Markov Model (HMM), pass through each word and gives the most likely tag [17]. Additionally, rather than searching a tag for each word separately, HMM explores a tag sequence for an entire sentence [18]. Consider that $\{w_1, w_2, \dots, w_n\}$ as the set of tokens, $\{t_1, t_2, \dots, t_n\}$ as the set of possible POS tags. Then, a HMM based tagger discovers the maximum probability of [18, eq. (1)].

$$P(t_1 \dots t_n, w_1 \dots w_n) = P(t_1 \dots t_n)P(w_1 \dots w_n | t_1 \dots t_n) \quad (1)$$

Finally, the sequence of $\{t_1, t_2, \dots, t_n\}$ which maximises (1) will be determined as a final POS tag.

2.4.3 Dependency Parsing

Dependency Parsing refers to examining syntactic grammatical dependency relation between the words of a sentence [19]. Not only that, but it can also merge tokens that had been over-segmented by the tokenizer previously.

As shown in [15, Fig. 4.], there are several direct links between every linguistic unit called dependencies. In detail, *nsubj* signifies that *This* is the subject due to the relationship with the VERB token, *is*.

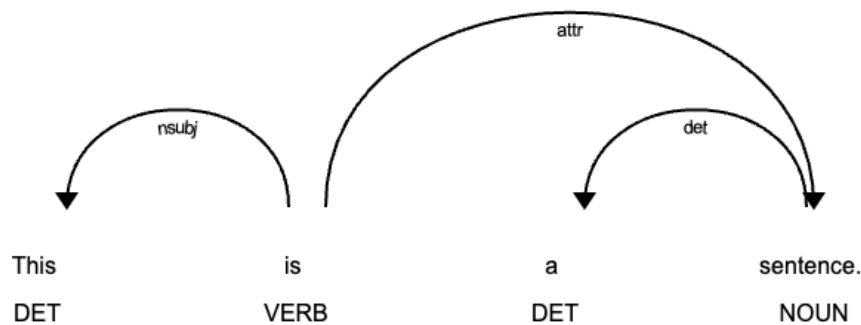


Fig. 4. Dependency parsing on POS tagged tokens

Moreover, the definition of dependency can denote as a directed graph $G = (V, A)$ where V represents nodes – tokens in this case, and A is arcs – relationships between tokens. When sentence $S = w_0 w_1 \dots w_n$ and R is a label set, the dependency graph holds the basic rule of [20, eq. (2)], [20, eq. (3)] and [20, eq. (4)]. Especially, (4) restricts more than one possible A between each pair of V , which G cannot be multi-digraphs. This definition of dependency graphs is unique to mono-stratal syntactic dependency theories, in which the entire dependency analysis is based on a single graph over the sentence's words [20].

$$V \subseteq \{w_0, w_1, \dots, w_n\} \quad (2)$$

$$A \subseteq V \times R \times V \quad (3)$$

$$if(w_i, r, w_j) \in A \text{ then } (w_i, r', w_j) \notin A \text{ for all } r' \neq r \quad (4)$$

2.4.4 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical method that determines how vital a word is in a collection of documents [21]. It is one of the vectorisation tools which converts text documents into numerical numbers so that the computer can compute essential words. TF-IDF score is calculated by [21, eq. (7)] when satisfies [21, eq. (5)] and [21, eq. (6)]. Equation (5) illustrates how term frequency calculates and (6) does inverse document frequency which indicates how frequent or uncommon a word is throughout all documents.

Therefore, if the word has a high frequency in the document, it will have a low TF-IDF value which designates low importance.

$$TF = \frac{\text{Total appearance of word in document}}{\text{Total words in a document}} \quad (5)$$

$$IDF = \log \frac{\text{All Document Number}}{\text{Document Frequency}} \quad (6)$$

$$TF - IDF = TF \times IDF \quad (7)$$

2.5 Classification Model

The most frequent implementation of multi-classifier system is multi-classification schemas based on parallel topology [22]. Especially, a recent study [7] has described that they have been extensively studied in the literature and well understood, moreover, if the basis classifiers are sufficiently independent, it increases their performance. Therefore, in this paper, different kinds of classifiers are analysed, such as Random Forest Classifier and four different Naïve Bayes Classifier – Gaussian Naïve Bayes, Multinomial Naïve Bayes, Bernoulli Naïve Bayes and Complement Naïve Bayes Classifier.

2.5.1 Random Forest Classifier

Random Forest (RF) refers to an ensemble learning method and generates a large number of individual decision trees which are uncorrelated to each other [2]. Then, as [43, Fig. 5.] expresses, the trees generated result in prediction value and RF selects the final prediction value which has the most counts among the results – as a prediction value of 1 has more counts than 0, the final prediction value is 1. Likewise, RF is based on the wisdom of crowds so that it reduces the possibility of getting biased results [2].

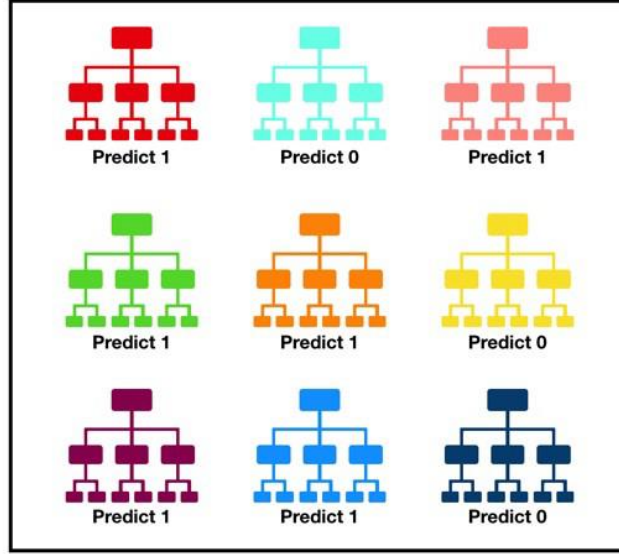


Fig. 5. Visualisation of Random Forest Classifier

2.5.2 Naïve Bayes Classifier

Naïve Bayes Classifier is a statistical classification algorithm based on the Bayes' theorem which provides the conditional probability of happening of two events based on the probabilities of given events as [23, eq. (8)] shows [23].

$$P(class|data) = \frac{P(data|class) \times P(class)}{P(data)} \quad (8)$$

According to a recent study [9], Naïve Bayes Classifier assumes that each attribute is independent from the others, which makes the algorithm faster and more sophisticated than other classifiers.

This paper will compare four different Naïve Bayes algorithms, Gaussian Naïve Bayes, Multinomial Naïve Bayes, Bernoulli Naïve Bayes and Complement Naïve Bayes Classifier.

2.5.2.1 Gaussian Naïve Bayes Classifier

Gaussian Naïve Bayes most commonly assumes Gaussian distributions by using only two values, mean and standard deviation [24]. This can be extended to the actual value attributes, called Gaussian Naïve Bayes. As it needs only two values, the calculation is easier than other Naïve Bayes Classifiers. The possibility is computed by [44, eq. (9)], assuming that if y is given class variable then consider $\{x_1, x_2, \dots, x_n\}$ as the dependent feature vector.

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \quad (9)$$

Where σ is the standard deviation for x , μ_y is the mean of the values in x associated with given class y .

2.5.2.2 Multinomial Naïve Bayes Classifier

Multinomial Naïve Bayes Classifier is compatible for discrete feature classification problems – such as word counts for text classification – and one of the two standard variants of Naïve Bayes for text classification [25]. It applies the Naïve Bayes algorithm to multinomially distributed data and requires integer feature counts mainly, but also works with TF-IDF [25]. Namely, the event of the multinomial algorithm represents the occurrence of a word in a single document and the likelihood is given by [44, eq. (10)].

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \quad (10)$$

Where θ_{yi} is the probability $P(x_i | y)$ when the distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \dots, \theta_{yn})$, $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T and $N_y = \sum_{i=1}^n N_{yi}$ is the total count of all features for class y .

In consequence, as Multinomial Naïve Bayes Classifier is a complete data mining approach, it is an adequate alternative for classifying a variety of data which is not easy to quantify.

2.5.2.3 Bernoulli Naïve Bayes Classifier

Although Bernoulli Naïve Bayes Classifier is preferred for document classification like Multinomial Naïve Bayes, it does not calculate the term frequency. Moreover, even multiple features are assumed as a binary value for data allocated by multivariate Bernoulli distributions, and the algorithm gives penalties to the non-occurrence of a feature i unlike the Multinomial algorithm as [44, eq. (11)] exhibits. Thus, Bernoulli is known for resulting in better performance on small text classification datasets than Multinomial [25].

$$P(x_i | y) = P(i | y)x_i + (1 - P(i | y))(1 - x_i) \quad (11)$$

2.5.2.4 Complement Naïve Bayes Classifier

Complement Naïve Bayes Classifier is deemed to supplement of Multinomial Naïve Bayes algorithm as multinomial algorithm easily get overfit with imbalanced dataset [26]. As seen in [25, eq. (12)], the Complement algorithm calculates the possibilities of an entity belonging to all classes rather than the possibilities of an entity belonging to a specific class. Equation (12) is the distribution rule when the weight is seen as [44, eq. (13)], where d_{ij} can be counts or TF-IDF value of term i in document j , α_i is smoothing hyperparameter also found in Multinomial Naïve Bayes and $\alpha = \sum_i \alpha_i$

$$\hat{c} = \arg \min_c \sum_i t_i w_{ci} \quad (12)$$

$$\begin{aligned} \hat{\theta}_{ci} &= \frac{\alpha_i + \sum_{j:y_j \neq c} d_{ij}}{\alpha + \sum_{j:y_j \neq c} \sum_k d_{kj}} \\ w_{ci} &= \log \hat{\theta}_{ci} \\ w_{ci} &= \frac{w_{ci}}{\sum_j |w_{cj}|} \end{aligned} \quad (13)$$

2.6 Related Work

The research are selected for discussing related technologies and approaches such as machine learning classifier specialised in text data, NLP analysis, character comprehension in English, and character profiling with literature.

- NLP Based Sentiment Analysis on Twitter Data Using Ensemble Classifiers

Kanakaraj et al. [2] proposed methods of classification to extract features from text data. Specifically, it uses several classifiers such as Naïve Bayes, Random Forest and so on, and tests its performance with Precision, Recall and F score. This study can help me identify classifiers and evaluation methods for my project. However, it uses a different data source than my project, which is SNS instead of literature, so the procedure of collecting the dataset differs.

- TVSHOWGUESS: Character Comprehension in Stories as Speaker Guessing

Sang et al. [3] provided a task of character comprehension with the script of the tv show. It proposed the way of collecting datasets with the rule-based parser, which is similar to the way of gathering datasets from literature. Also, their dataset is written in English as well. Nevertheless, it uses deep learning classifiers so the process of evaluation would be different.

- Speaker Extraction

Pang [27] presented the source code for extracting the speaker from a Chinese novel. It suggested how to split the document into dialogue and speaker, which helps me to collect the dataset from raw documents. Nonetheless, the dataset is written in Chinese, so the grammar and the dialogue rule are different from English ones. It is essential to consider grammar and build different functions.

Summary

With the growth of very successful machine learning (ML) over the last two decades, natural language processing (NLP) has progressed significantly together. In particular, character

profiling is the field that NLP has been investigated frequently with different data. However, this project focuses on implementing profiling with a special source which is literature. The biggest challenge will be collecting datasets from the raw document because it is hard to extract speakers and dialogue. To overcome this, NLP techniques will be applied using SpaCy's tokenizer, tagger and parser. Especially, SpaCy has high performance on POS tagging so it is expected to derive high accuracy for this project.

3. Design

This chapter will explain the design decisions and the tool used to achieve the final implementation of the project.

3.1 Plan

Since the limited time is given to complete the project, the most efficient methodology, the agile method has been conducted. This methodology has the structure that each stage is repeated until the final launch as seen in [45, Fig. 6.] Unlike traditional methodology, agile methodology can reduce the risk and raise the quality of the project by getting tests and feedback after any development. As the regular meetings are held with my supervisor every week, it was easy to get feedback and review the project every time there is progress.

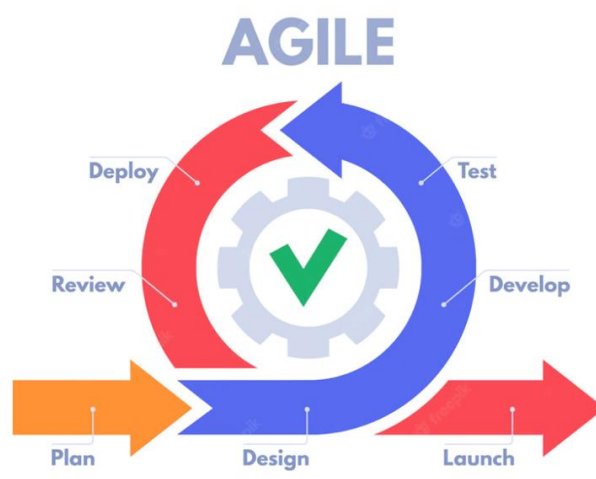


Fig. 6. Agile methodology

3.2 Tool

3.2.1 Python

Python is a dynamically semantic, interpreted, object-oriented and high-level programming language. Its high-level built-in data structures, together with dynamic typing and dynamic binding, make it ideal for Rapid Application Development and as a scripting or glue language for connecting existing components. Python's concise and easy-to-learn syntax promotes readability, which lowers software maintenance costs. Python facilitates programme flexibility and code reuse by supporting modules and packages. The Python interpreter and its substantial standard library are free to download and distribute in source or binary form for all major platforms. [28]

The edit-test-debug cycle is extraordinarily rapid because there is no compilation step. Python scripts are simple to debug: a bug or improper input will never result in a

segmentation fault. Instead, when the interpreter finds a mistake, it throws an exception. The interpreter prints a stack trace if the application fails to catch the exception. Inspection of local and global variables, execution of arbitrary expressions, setting breakpoints, stepping through the code one line at a time, and so on are all possible with a source level debugger. The debugger is written in Python, demonstrating Python's introspective capabilities. On the other hand, adding a few print statements to the source code is frequently the quickest method to debug a programme: the fast edit-test-debug cycle makes this simple approach quite successful. [28]

3.2.2 Google Colab

Google Colab is a web IDE for python created by Google. It is based on the Jupyter Notebook and enables machine learning, data analysis and education on the web. Technically, it can utilise Jupyter Notebook without setup as well as provides free GPUs. [29]

However, the free version of Google Colab has a short run time and limited GPU. For example, if you run huge data the run time can crash and stop the programme. Also, as it uses Google drive, it is hard to work with a huge dataset over 15GB. [29]

3.2.3 Scikit-learn

Scikit-learn is a machine learning toolkit for Python. It includes packages for classification, regression and predictive analytics. Especially, all the classifiers that I used for my project are included in this library.

3.2.4 SpaCy

SpaCy is a Python open-source library for Natural Language Processing (NLP) that includes a variety of packages for data processing and analysis. To extract useful information from unstructured textual data, NLP can assist to represent the data in a format that the computer can understand, and the method used for this process is SpaCy.

4. Implementation

This chapter will describe the details of what development was done and the reason why. Also, it will discuss what methodologies are implemented and the limitations and the strengths of them.

Overview

The project is developed following the process shown in Fig. 7. The whole process separates into 5 big stages, starting from Collecting Dataset to Evaluation. As Fig. 7. shows, the biggest challenge of this project is collecting datasets.

The stage of collecting the dataset mainly focuses on NLP and a rule-based parser is produced to extract context, dialogue and speaker. After collecting the dataset successfully, pre-processing is applied using vectorisation. Then, to get the highest performance, hyper-parameter tuning is implemented to all classifiers. After that, the machine is trained by the set of dialogue and speaker data, and recognises the feature of dialogues from the specific speaker, which is the processing of profiling. In addition, analysis is implemented with the training by comparing the results from different classifiers and datasets in the evaluation stage. This section will the technologies used for each process and explains the reason.

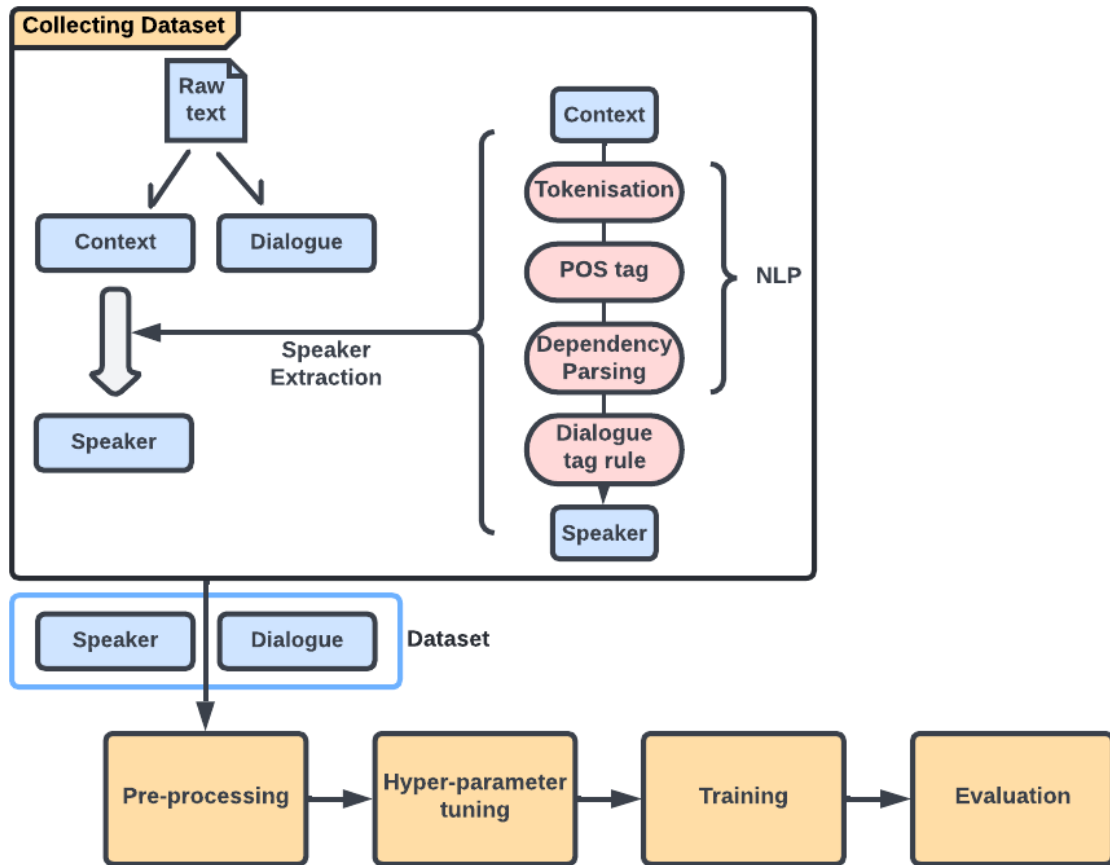


Fig. 7. Entire process of the project

4.1 Data Collection

4.1.1 Dialogue & Context Extraction

Before training the data, collecting labelled data is necessary – which consists of dialogues and speakers – from the literature document. Initially, as the literature document is a raw text, it is split into the corpus of context and dialogue as shown in Fig. 8. In Fig. 8., column *istart* and *iend* is the index of the beginning and the end of the dialogues, and the *talk* is the content of the dialogue, and the *context* is the sentences after the dialogue until the next one.

	istart	iend	talk		context
0	3614	3666	Whatever have you been doing with yourself, Wa...	he asked in undisguised wonder, as we rattled ...	
1	3750	3799	You are as thin as a lath and as brown as a nut.	I gave him a short sketch of my adventures, an...	
2	3920	3932	Poor devil!	he said, commiseratingly, after he had listen...	
3	4001	4025	What are you up to now?		
4	4295	4318	And who was the first?		I asked.

Fig. 8. Dataset of dialogue and context

To split the document shown in Fig. 8., various approaches are conducted, such as NLTK and Regular Expression. However, both approaches only remain one sentence in the *talk* column as it is splitting every sentence instead of detecting quotation marks. Thus, the parser is built as a user-defined function called *get_conversations()* as seen in Fig. 9.

Basically, the parser consists of two large parts, extracting dialogues and extracting context. Once, the document is obtained, it regards the whole document as a *string* type, and breaks the *string* into *character* types so that the iteration can detect each character one by one. Then, it obtains the dialogues by finding an index of a specific *character* that matches two quotation marks which are the opening quotation mark and closing quotation mark.

After that, it extracts the compatible context for the dialogue. Normally, the context follows the end of the dialogue but sometimes it locates at the front of the dialogue. For example, if the previous context is ended with *remarked;*, then that context is compatible with both the previous dialogue and the next dialogue. Thus, the parser consists of a condition statement considering different cases.

```
def get_conversations(sentence):
    istart, iend = -1, -1
    talks = []
    # Extract Dialogues
    for i in range(len(sentence)):
        if sentence[i] == '"':
            istart = i
            for j in range(i, len(sentence)):
                if (sentence[j] == '"' and (not istart == -1)):
                    iend = j
                    conversation = {'istart':istart, 'iend':iend, 'talk':sentence[istart+1:iend].replace('\n', ' ')}
                    talks.append(conversation)
                    istart = -1
    contexts = []
    # Extract Context
    for i in range(len(talks)):
        if i+1 >= len(talks): # for the last index
            contexts.append(sentence[talks[i]['iend']+1:len(sentence)])
        elif sentence[talks[i]['istart']-2] == ';': # for remarked;
            contexts.append(sentence[talks[i-1]['iend']+1:talks[i]['istart']])
        else: # for normal case
            contexts.append(sentence[talks[i]['iend']+1:talks[i+1]['istart']])
        talks[i]['context'] = contexts[i]
```

Fig. 9. Code of *get_conversations()*

4.1.2 Natural Language Processing

As *context* corpus is acquired in the dataset, it is possible to identify the *speaker* by analysing the *context* according to the dialogue tag rule. However, before analysing, natural language processing is required to let the computer recognise the corpus.

In practice, NLP is implemented by the tool. It has several state-of-the-art libraries such as Natural Language Toolkit (NLTK) and SpaCy. Both of them are open-source libraries based on Python and include the package for tokenisation, POS tagging and dependency parsing but it has the difference in approach.

NLTK

NLTK was established in 2001 as part of a computational linguistics course at the University of Pennsylvania's Department of Computer and Information Science. With the support of many contributors, it has been built and expanded since then. It is now used in dozens of university courses and provides the foundation for numerous research projects [30].

Moreover, NLTK has numerous capabilities, including text cleaning, word form merge, part of speech tagging, grammatical analysis, and semantic analysis. Researchers can use this toolkit to complete the entire process from corpus creation to research retrieval in one environment, minimising the hassle of switching between different software and data conversion, and improving the scope and depth of their research [31].

SpaCy

SpaCy is a relatively new addition to the Python text processing environment, but it is a fairly steady framework. It focuses on industrial-strength natural language processing, hence it is suitable for large-scale corpus text analytics. It accomplishes this efficiency by making use of Cython's memory-managed operations [32].

Furthermore, SpaCy also includes the linguistic data and techniques required to process natural language texts like NLTK. As it has container objects that represent natural language text elements like sentences and words, it is convenient to use. The properties of these objects indicate linguistic features in turn such as POS tag. Additionally, it includes built-in visualisers that users can use to build a graphic of a sentence's syntactic structure or named entities in a document [33].

Practically, NLTK is an open-source library that is widely used by linguists and researchers, and it is a string processing library – taking input as string type and returning output as a string. On the other hand, SpaCy has a different aspect of design decisions. While NLTK is research software, SpaCy is for completing tasks.

However, there has been a study [14] which compares the performance of several popular libraries including NLTK and SpaCy. It has been conducted by different sectors such as tokenisation and POS tagging with different data sources. Although the test results are different depending on the dataset, NLTK has shown high performance on tokenisation and SpaCy has the most promising accuracy on POS tagging generally. In consequence, Omran and Treude [14] proposed that choosing the appropriate NLP library can differ by task and source.

Since my project focuses on word tokenisation and POS tagging, SpaCy is chosen as an NLP tool as it is well known for high performance on both of them, as well as processing huge amounts of data quickly [34]. Thus, using SpaCy would be beneficial when implementing NLP on literature.

With the corpus that we got from *4.1.1 Dialogue & Context Extraction*, SpaCy's tokenizer tokenised the *context* corpus into words/tokens, in the way *2.4.1 Tokenisation* describes. After that, every token got POS tagging to categorise its attributes as explained in *2.4.2 POS tagging*. Then, the dependency parser examined each token's dependency according to the relationship with other tokens in the sentence as elucidated in *2.4.3 Dependency Parsing*.

4.1.3 Dialogue Tag Rule

To keep the high quality of writing, writers follow the rule of dialogue when writing literature, which can use to build a function for extraction of the speaker from context.

For instance, the basic rule of the dialogue tag is mentioning the speaker with the action beat such as ‘he said’ or ‘said Stamford’ as you can see from Fig. 10.

In this case, ‘he said’ follows the order of *subject* and *verb*, and this can be deemed as the *subject* is considered a speaker in this sentence structure. In other words, the dialogue, which has the context whose first token’s dependency label is *nsubj* and the second token’s POS tag is VERB, will be assigned a *subject* in the speaker dataset (Fig. 11.). Likewise, ‘said Stamford’ will be detected by the feature that the POS tags of the first two tokens are in order of VERB and PROPN (Fig. 11.).

“Dr. Watson, Mr. Sherlock Holmes,” **said Stamford**, introducing us.

“How are you?” **he said** cordially, gripping my hand with a strength for which I should hardly have given him credit. **“You have been in Afghanistan, I perceive.”**

“How on earth did you know that?” **I asked** in astonishment.

“Never mind,” **said he**, chuckling to himself. **“The question now is about hoemoglobin. No doubt you see the significance of this discovery of mine?”**

“It is interesting, chemically, no doubt,” **I answered**, “but practically---”

Fig. 10. Case of dialogue tagging 1

```
if tokenlist and tokenlist[0].dep_ == "nsubj" and tokenlist[1].pos_ == "VERB":
    speaker = tokenlist[0]
    talk['speaker'] = speaker
elif tokenlist and tokenlist[0].pos_ == "VERB" and tokenlist[1].pos_ == "PROPN":
    speaker = tokenlist[1]
    talk['speaker'] = speaker
```

Fig. 11. Code of extracting speaker with NLP

On the other side, according to [35], if the same speaker is talking in the next dialogue, the next one will follow the current context without a line break to avoid repetition, this can be seen in highlighted sentences in Fig. 10. and Fig. 12.

However, since the Fig. 10. and the Fig. 12. differ in the existence of following contexts – while the highlighted dialogues in Fig. 10. has no following context, the dialogue in Fig. 12. has the compatible context followed, it is significant to build two independent functions for each case to collect more dataset.

“Whatever have you been doing with yourself, Watson?” **he asked** in undisguised wonder, as we rattled through the crowded London streets. **“You are as thin as a lath and as brown as a nut.”**

I gave him a short sketch of my adventures, and had hardly concluded it by the time that we reached our destination.

Fig. 12. Case of dialogue tagging 2

In another case, one of the basic rules of writing is the Three-Beat rule, which guides writers to keep the three dialogue beats as a maximum [35]. As Fig. 13. shows, highlighted dialogue has no context and speaker tag, nevertheless, since it follows the three-beat rule we could presume easily that the speaker is *my companion* in this case. This is a situation in which the characters speak alternately, so it can be acknowledged that the speaker is the same as the one twice before.

“That’s a strange thing,” remarked my companion; “you are the second man to-day that has used that expression to me.”

“And who was the first?” I asked.

“A fellow who is working at the chemical laboratory up at the hospital. He was bemoaning himself this morning because he could not get someone to go halves with him in some nice rooms which he had found, and which were too much for his purse.”

Fig. 13. Case of dialogue tagging 3

After completing extraction based on the dialogue tag rule, the result looks like Fig. 14. As I described previously, the dialogue may not possess compatible context, due to the rule of tagging dialogue.

talk	context	speaker
Whatever have you been doing with yourself, Wa...	he asked in undisguised wonder, as we rattled ...	he
You are as thin as a lath and as brown as a nut.	I gave him a short sketch of my adventures, an...	he
Poor devil!	he said, commiseratingly, after he had listen...	he
What are you up to now?		he
And who was the first?	I asked.	I

Fig. 14. Dataset of dialogue, context and speaker

However, because this dialogue tag rule is a recent one, there may be other types of literature that use other dialogue tag rules such as old literature. The datasets picked for this project follow the same dialogue tag rule, however, if other literature followed different rules wants to be evaluated, it may produce issues while extracting information.

4.2 Coreference Resolution

Although the dataset is collected from raw text successfully, most of the value in the *speaker* column in Fig. 14. is the pronoun – he or I – instead of the character’s name. Even though the speaker ‘I’ represents the narrator of the book, a single person, the speaker ‘he’ could represent more than two characters. This can occur ambiguity of each character’s features, which might lead to the reduction in accuracy of the training result. Hence, the Coreference Resolution approach has been conducted to replace pronouns with characters’ names.

Coreference Resolution is the task of finding the same entity among multiple sentences in natural language. Specifically, Neuralcoref framework has been applied as it is based on

SpaCy and appeared to have state-of-the-art accuracy. Also, it consists of two sub-modules, “a rule-based mentions-detection module which uses SpaCy's tagger, parser and NER annotations to identify a set of potential coreference mentions, and a feed-forward neural-network which compute a coreference score for each pair of potential mentions.” [36]

As Fig. 15. describes, every entity gets its dependency label such as pronominal and nominal, and the coreference score is computed between each entity. The higher the score, the more likely it is that the two entities are the same. In particular, the entity *My sister* and *She* have the highest score of 7.66, which represents that they are the same entity. On the contrary, the lowest score of -3.37 appears in the relation between the entity *My sister* and *a dog*, which seems less relative.

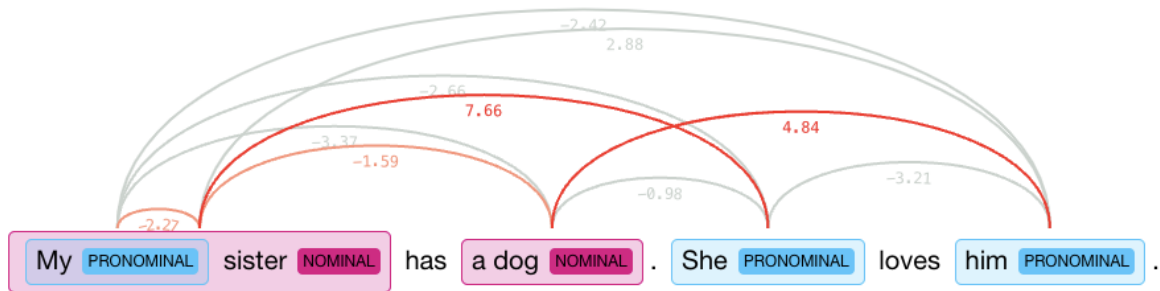


Fig. 15. Score of coreference resolution

Nonetheless, regardless of its proven accuracy, the accuracy of the literature dataset was low. It is suspected that this is because Neuralcoref has higher performance with short text samples instead of large data. As literature often manifests who the speaker is by implying their behaviour, there could be no mentions of the speaker’s name frequently, which means there might be no entity of the character’s name in the range of relation detection. Even though the parameter values have been raised to regulate the range of detecting, the accuracy has not increased distinctively. Interestingly, the accuracy stops increasing at a certain point when the parameter values are continued to rise.

Therefore, instead of implementing coreference resolution, the pronoun is converted to the character’s name based on its probability like Fig. 16.

talk	context	speaker
Whatever have you been doing with yourself, Wa...	he asked in undisguised wonder, as we rattled ...	Sherlock Holmes
You are as thin as a lath and as brown as a nut.	I gave him a short sketch of my adventures, an...	Sherlock Holmes
Poor devil!	he said, commiseratingly, after he had listen...	Sherlock Holmes
What are you up to now?		Sherlock Holmes
And who was the first?	I asked.	John Watson

Fig. 16. Final dataset

4.3 Pre-processing

Since the computer cannot understand natural language, applying vectorization is necessary before training the text data. First, *CountVectorizer* is implemented to convert text into the vector of term counts. Yet, having counts of words may not differentiate different characters' features – some terms are repeated without any meaning, *TF-IDF vectorization* is applied to increase accuracy since it weighs the words based on their frequency.

4.3.1 TF-IDF Vectorization

TF-IDF is implemented to analyse the character's features through their dialogue corpus. It computes the significance of each token in the corpus and scores them with numerical values as shown in Fig. 17. TF- IDF is selected to execute vectorization as it has demonstrated its accuracy in various NLP sectors as seen from recent studies [37] and [38]. As explained in 2.4.4 *Term Frequency-Inverse Document Frequency*, the word possesses a higher frequency indicates a lower word weight, which means word *the* in Fig. 18. has the lowest word weight as it is common in the corpus, on the other hand, *readiness*, *room* and *open* have the higher word weight as they are unique.

Nonetheless, TF-IDF has several drawbacks. For example, it might be slow with huge datasets since it calculates similarity with the frequency directly. Also, it is presumptively assumed that the counts of various terms give independent evidence of similarity.

idf_weights	
the	2.066351
you	2.166435
to	2.166435
it	2.220502
and	2.338285
...	...
ineffable	4.417727
individuals	4.417727
indeed	4.417727
ill	4.417727
yours	4.417727

Fig. 17. TF-IDF scores of John Watson's corpus

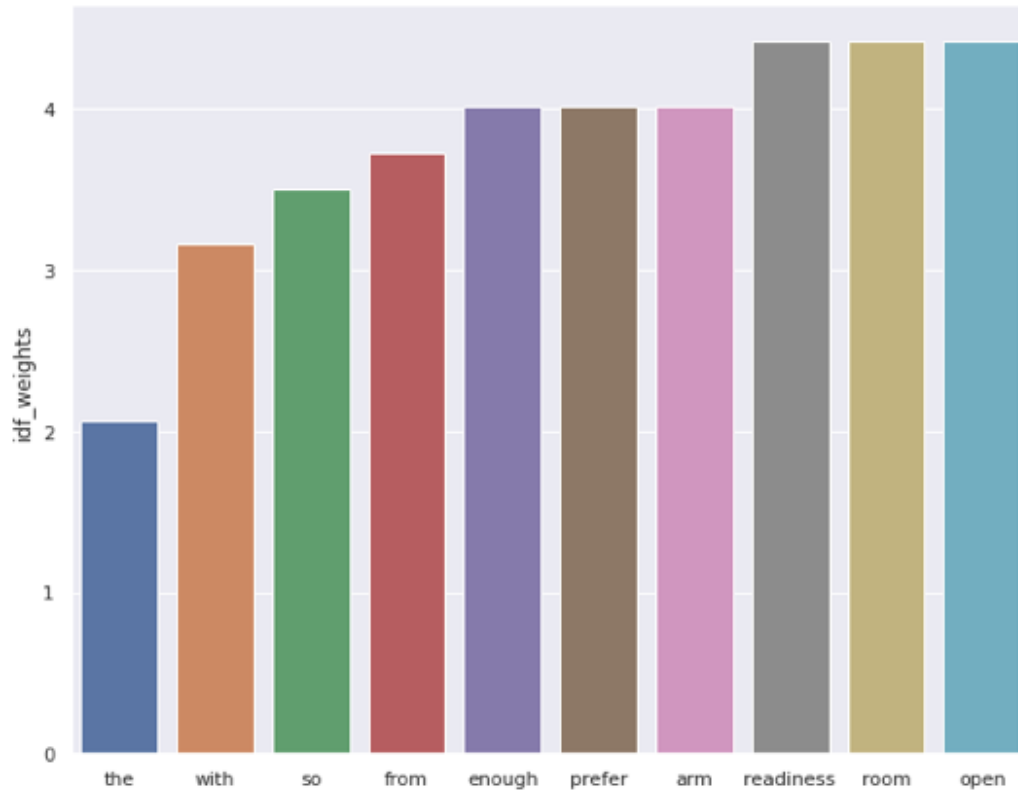


Fig. 18. Word weight of John Watson's corpus

4.4 Hyper-parameter tuning

Since text distribution is classification problem, Random Forest Classifier and 4 different Naïve Bayes Classifier – GaussianNB, BernoulliNB, MultinomialNB, ComplementNB – are used for training. As I described in 2.5 *Classification Model*, they have evinced their high capabilities with text classification. Moreover, hyper-parameter tuning has been carried out to attain the highest accuracy and GridSearchCV tool is used which is based on the cross-validation score.

4.4.1 GridSearchCV

GridSearchCV adjusts parameters to determine the optimal values to find the best parameters [39]. It is a function provided by Scikit-learn and uses the cross-validation method to evaluate the model with various parameters [40]. Specifically, GridSearchCV fits all required models with different parameter values as shown in Fig. 19. and returns the best parameters like Fig. 20. After that, it can also train the model with the best parameters obtained after hyper-parameter tuning. As such, hyper-parameter tuning is completed, and all models are trained with the best output. The result will be compared in 5.2.2 *Hyper-parameter tuning*.

However, one of the weaknesses of GridSearchCV is that it struggles with dimensionality when the number of hyper-parameters increases dramatically. In other words, it is computationally intensive because it becomes more complicated as the number of parameters in the param grid increases. As a result, the GridSearchCV approach is not suggested for the large datasets or parameter grids with many components.

```

Grid scores on development set:

0.667 (± 0.078) for {'max_depth': 5, 'n_estimators': 10}
0.682 (± 0.208) for {'max_depth': 5, 'n_estimators': 50}
0.702 (± 0.223) for {'max_depth': 5, 'n_estimators': 100}
0.694 (± 0.150) for {'max_depth': 5, 'n_estimators': 200}
0.702 (± 0.164) for {'max_depth': 10, 'n_estimators': 10}
0.688 (± 0.072) for {'max_depth': 10, 'n_estimators': 50}
0.701 (± 0.136) for {'max_depth': 10, 'n_estimators': 100}
0.701 (± 0.154) for {'max_depth': 10, 'n_estimators': 200}
0.715 (± 0.091) for {'max_depth': 20, 'n_estimators': 10}
0.701 (± 0.119) for {'max_depth': 20, 'n_estimators': 50}
0.694 (± 0.137) for {'max_depth': 20, 'n_estimators': 100}
0.702 (± 0.223) for {'max_depth': 20, 'n_estimators': 200}
0.674 (± 0.093) for {'max_depth': 50, 'n_estimators': 10}
0.674 (± 0.195) for {'max_depth': 50, 'n_estimators': 50}
0.681 (± 0.116) for {'max_depth': 50, 'n_estimators': 100}
0.701 (± 0.092) for {'max_depth': 50, 'n_estimators': 200}

```

Fig. 19. Hyper-parameter tuning with GridSearchCV on Random Forest Classifier

```

Best parameters set found on development set:

{'max_depth': 20, 'n_estimators': 10}
RandomForestClassifier(max_depth=20, n_estimators=10)
0.7152709359605912

```

Fig. 20. Best parameters set on Random Forest Classifier with GridSearchCV

4.5 Training

As illustrated in *2.5 Classification Model*, five diverse classifiers from the Scikit-learn library have been performed for the training model. The major explanation is already described in *2.5 Classification Model*, this section will mainly discuss the strengths and limitations of each classifier.

4.5.1 Random Forest Classifier

As Random Forest (RF) produces a large number of trees that are independent from each other, it helps to improve decision tree accuracy by reducing overfitting. Also, since it uses a rule-based approach, no data normalisation is necessary.

Despite these benefits, a random forest method has certain disadvantages as well. As it creates several trees and combines their outputs, it necessitates a lot of computer power and resources. In addition, since it uses a number of decision trees to select the class, it takes a long time to train.

4.5.2 Gaussian Naïve Bayes Classifier

As explained in 2.5.2.1 *Gaussian Naïve Bayes Classifier*, it has the simplest algorithm among Naïve Bayes Classifier, it takes less time to train. In addition, if the premise of feature independence remains true, it can outperform other models while using far less training data. Moreover, it works well with categorical input variables rather than numerical ones.

However, it has also had critical disadvantages depending on the problem. For example, it is using conditional independence in the algorithm, but not every problem has conditional independence assumption. Furthermore, if some words exist in test data but not in training, the result becomes zero, which is called zero class probability.

4.5.3 Bernoulli Naïve Bayes Classifier

Bernoulli Naive Bayes produces more accurate and exact answers than others when dealing with little amounts of data or small documents like text classification. Also, it can effectively handle irrelevant features by ignoring the absent features, while other models are penalising them. Thus, it can save time on training.

Nevertheless, it also has some limitations. For instance, if there are interdependent features in the problem, it can reduce the accuracy and influence the prediction result. Besides, it can also lead to zero class probability like Gaussian Naïve Bayes.

4.5.4 Multinomial Naïve Bayes Classifier

Unlike other machine learning (ML) approaches used for content-based text classification, Multinomial Naïve Bayes enables the model to predict classes for texts without continuous training since it is a data mining strategy. Thus, it can be a powerful alternative to the well-known ML approaches.

The disadvantage of Multinomial Naïve Bayes is it can occur zero class probability as its algorithm is based on the Gaussian Naïve Bayes.

4.5.5 Complement Naïve Bayes Classifier

As explained in 2.5.2.4 *Complement Naïve Bayes Classifier*, it is derived from Multinomial Naïve Bayes and handles imbalanced data sets well. Also, it increases its performance by computing the model's weights using the complement of each class even though it is using the class for which the conditional probability. Therefore, its parameter estimations are more stable and outperform text classification tasks than the Multinomial algorithm.

However, just like other Naïve Bayes Classifiers, as it is depending on the conditional independence assumption, it can result in zero class probability.

4.6 User Interface

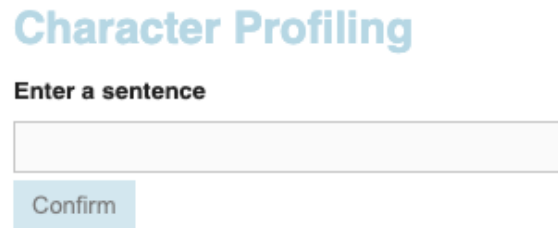
To visualise the result of the project, a user interface (UI) is built. However, there have been problems since Google Colab has no interaction with the Web Application Tool. Thus, Ipywidget is utilised which allows building UI on Jupyter Notebook.

4.6.1 Ipywidget

According to [41], ipywidget can embed JavaScript and HTML in Jupyter Notebook and IPython kernel. When interactive widgets are employed, users have control over their data

and can see how it changes. Unlike other web application tools, it is extremely easy to use because you can design UI without set-up. However, there is limited functionality that can be used. Also, it is hard to organise the front end and back end as all the code is written in the same cell.

As shown in Fig. 21., the random sentence will enter by the user. And it will show the speaker's name by guessing who likely said the input sentence. If the machine profiles the characters successfully, then it will guess right.



The image shows a web interface titled "Character Profiling" in a light blue font. Below the title, there is a label "Enter a sentence" in a bold black font. Underneath the label is a light gray rectangular text input field. Below the input field is a light blue button with the word "Confirm" in white text.

Fig. 21. UI for guessing character's name

5. Result and Evaluation

This chapter will cover the discovery from the project, providing the evidence in the form of line plots, bar graphs, confusion matrix, figures and tables. Especially, F1-score, cross-validation score, recall and precision values are produced to assess the prediction's accuracy. It will discuss the dataset used and compare the result between the original model and the ones applying coreference resolution, also analyse the difference before and after adapting hyper-parameter tuning. Besides, the confusion matrix and performance metrics are produced to assess the best model and examine the contrast between the two datasets.

Testing Environment

For most of the project's execution, Google Colab's 12GB RAM is enough. However, when applying coreference resolution with huge data, Google Colab always crashes due to the lack of RAM. As a result, Google Colab Pro is selected as a final testing environment which includes three different GPUs, Nvidia K80, T4 and P100 and more RAM of 32GB. However, without coreference resolution, the environment of Google Colab is enough to run my project. This is the detail of the specification for the testing environment:

Google Colab

- GPU: Nvidia K80
- RAM: 12GB
- Runtime: 12 hours

Google Colab Pro

- GPU: Nvidia K80, T4 and P100
- RAM: 32GB
- Runtime: 24 hours

5.1 Dataset

It helps to investigate more precise models that comparing the result of two or more datasets rather than the result from only one dataset. Therefore, the datasets are collected from two different kinds of literature which are well known as the greatest books. Since this project focuses on dialogue datasets, it is important to select the book which includes the greatest dialogue, as this research [42] said. Thus, *A Study in Scarlet* by Arthur Conan Doyle and *Jane Eyre* by Charlotte Brontë are chosen.

Additionally, as the number of characters that appear in each chapter of the book is different, the chapters, which have the same character, have been selected for the persistence of the training.

In this paper, *A Study in Scarlet* by Arthur Conan Doyle is named as dataset A and *Jane Eyre* by Charlotte Brontë as dataset B. However, since both Fig. 22 (a) and Fig. 22 (b) have imbalanced datasets, which can lead to poor performance, up-sampling has been applied with Synthetic Minority Oversampling Technique (SMOTE) techniques to balance each count of character's data. As a result, dataset A becomes balanced data of *Sherlock Holmes*: 60, *John Watson*: 60, *Stamford*: 60 and dataset B becomes *Jane Eyre*: 129, *Mr. Rochester*: 129.

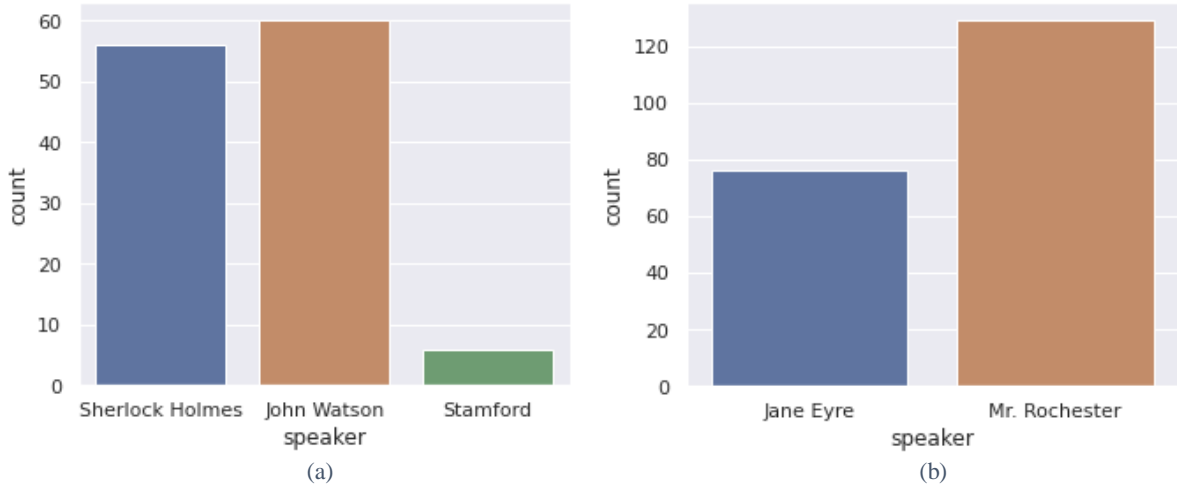


Fig. 22. Amount of data before SMOTE. (a) dataset A. (b) dataset B

5.2 Evaluation

5.2.1 Coreference Resolution

As illustrated in 4.2 *Coreference Resolution*, Neuralcoref has been applied to convert pronouns into characters' names. However, Neuralcoref has shown low accuracy despite its proven accuracy.

Fig. 23. and Fig. 24. show the comparison before and after adapting coreference resolution and the precise values are visible in Appendix A. As they show, the results are decreased with every classifier and for every dataset. Especially, Complement Naïve Bayes shows the lowest accuracy for both datasets regardless of the original model's accuracy.

It is suspected that it occurs since Neuralcoref recognises too many clusters not only the character's name but also the place's name. This is because when Neuralcoref is collecting the clusters, it is based on the POS tag which is *proper noun*, so it can be a name, part of the name, unique entity, place or object such as England or UN.

Also, as explained in 4.2 *Coreference Resolution*, since literature seldom mentions character's name, it is hard to match the pronoun with the named entity due to the long distance between the two entities.

Thus, due to the low performance of Neuralcoref, the project has moved on without applying coreference resolution.

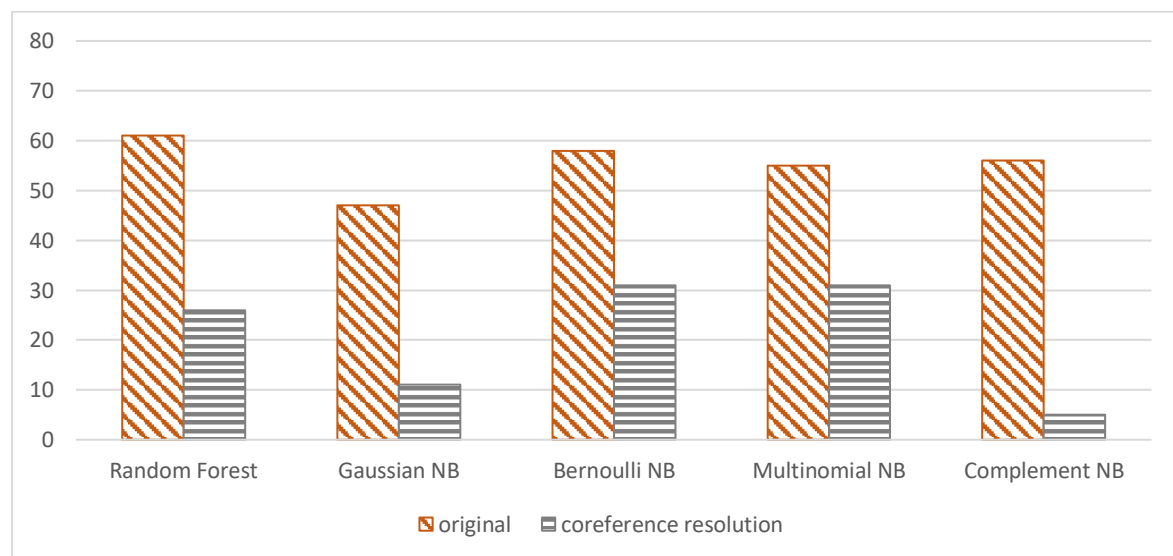


Fig. 23. Comparison 'with and without coreference resolution' on dataset A

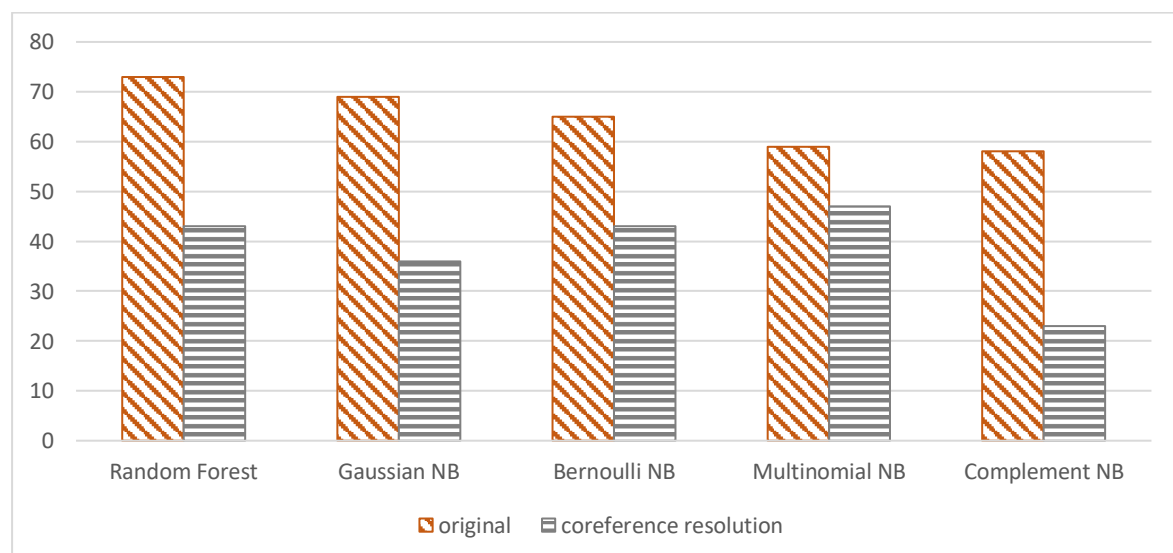


Fig. 24. Comparison 'with and without coreference resolution' on dataset B

5.2.2 Hyper-parameter tuning

As described in 4.4 *Hyper-parameter tuning*, GirdSearchCV has been implemented for 5 different classifiers with dataset A and B.

Fig. 25. and Fig. 26. are the comparison of F1-score before and after hyper-parameter tuning for both datasets and the detailed values are visible in Appendix B. As they show, generally hyper-parameter tuning increases the model's accuracy. Nonetheless, some classifier shows lower accuracy than before implementation.

It is interesting that the result is different for dataset A and B. For example, whilst the classifier, that shows the decreasing result, is Bernoulli Naïve Bayes with dataset A, it is Random Forest with dataset B. Also, the accuracy was the same for Complement Naïve Bayes on dataset A and Gaussian Naïve Bayes on dataset B. Probably, it is because Random Forest picks the data randomly every run of predictions, so the result has a slight difference every time.

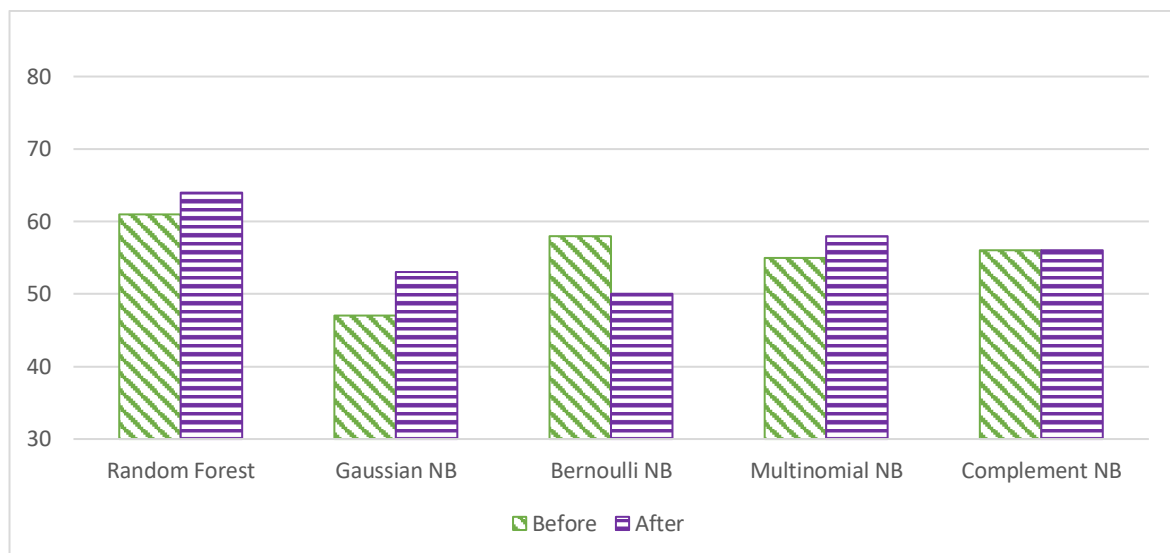


Fig. 25. Comparison before and after hyper-parameter tuning on dataset A

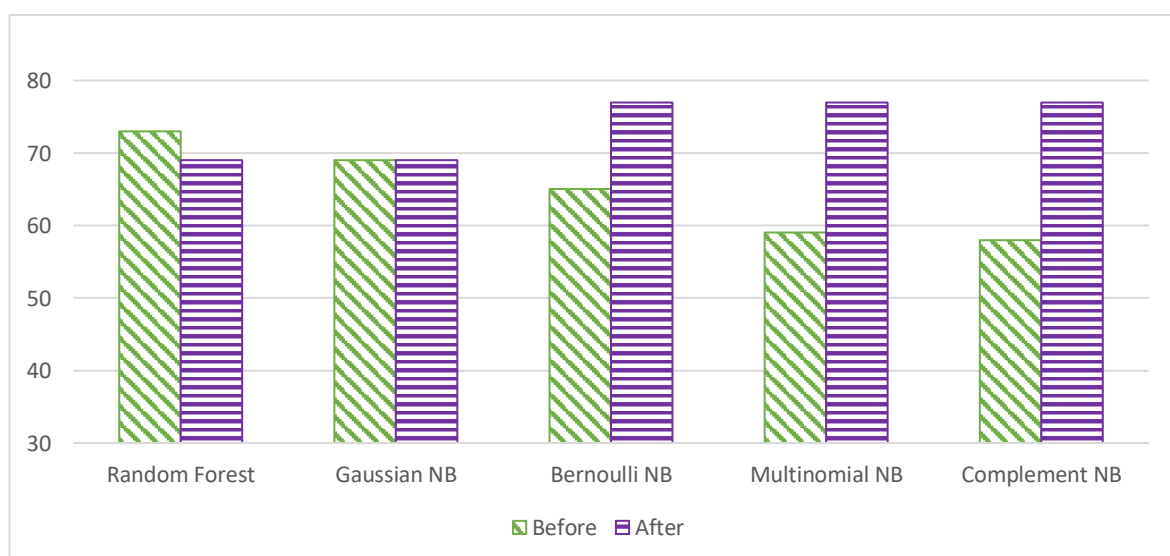


Fig. 26. Comparison before and after hyper-parameter tuning on dataset B

Table 1 and Table 2 are exhibiting the details of the outcomes after hyper-parameter tuning such as the value of the best parameters, F1-score and cross-validation score. As described in 4.4 *Hyper-parameter tuning*, the cross-validation score is the score of the GridSearchCV.

Generally, the cross-validation score shows higher accuracy than F1-score and dataset B shows higher accuracy than dataset A even though they have experimented on the same model. The details are compared in the next paragraph with the line plots.

TABLE I
Best Result of Hyper-Parameter Tuning on Dataset A

	Best Parameters	F1-score	Cross-validation score
Random Forest Classifier	max_depth: 50, n_estimators: 200	0.64	0.73
Gaussian Naïve Bayes	var_smoothing: 0.001	0.53	0.72
Bernoulli Naïve Bayes	alpha: 0.01	0.50	0.78
Multinomial Naïve Bayes	alpha: 0.01	0.58	0.72
Complement Naïve Bayes	alpha: 0.01	0.56	0.71

TABLE II
Best Result of Hyper-Parameter Tuning on Dataset B

	Best Parameters	F1-score	Cross-validation score
Random Forest Classifier	max_depth: 20, n_estimators: 200	0.69	0.77
Gaussian Naïve Bayes	var_smoothing: 0.01	0.69	0.74
Bernoulli Naïve Bayes	alpha: 0.01	0.77	0.70
Multinomial Naïve Bayes	alpha: 0.01	0.77	0.79
Complement Naïve Bayes	alpha: 0.01	0.77	0.78

Fig. 27. and Fig. 28. are plotted to compare the average of each score – F1-score and cross-validation score and their precise values are visible in Appendix C and Appendix D. The important result is that both of them are proving the improvement in performance with the average value of different datasets after hyper-parameter tuning.

Overall, as shown in Fig. 27. and Fig. 28., the cross-validation score is usually higher than F1-score with both dataset A and B. Also, dataset B reaches a higher score than dataset A in every kind of score. F1-score seems to differ by more than 10 percent depending on the dataset, while the cross-validation score does not have much difference with any data. Nevertheless, Random Forest Classifier appears to have a relatively stable F1-score no matter what the dataset is. Also, its average accuracy is high in both F1-score and cross-validation scores. Furthermore, Multinomial NB and Complement NB are also high in the average F1-score in Fig. 27.

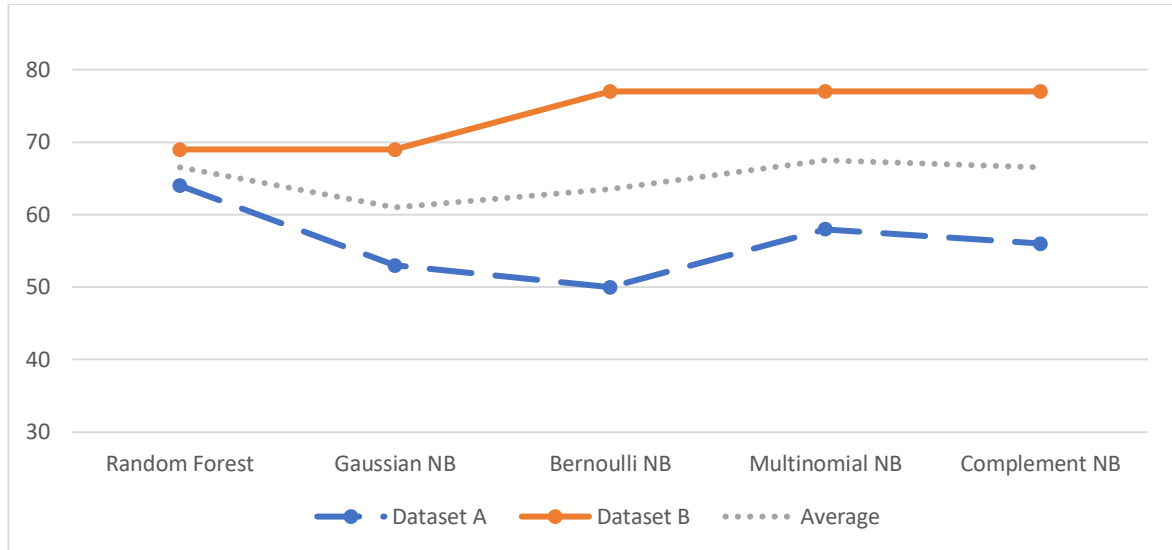


Fig. 27. F1-score comparison between dataset A and dataset B

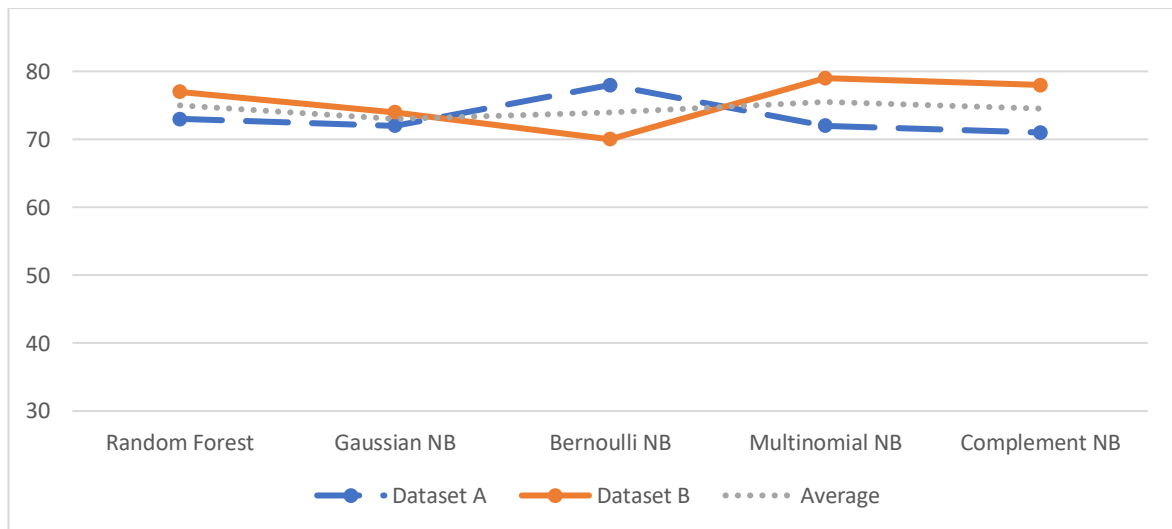


Fig. 28. Cross-validation score comparison between dataset A and dataset B

5.2.3 Confusion Matrix

To visualise the prediction result, the confusion matrix is plotted. It will show the number of correct and incorrect predictions among the classes. The x-axis represents the predicted values and y-axis are the actual values. Fig. 29. and Fig.30. are the confusion matrix for every five classifiers. For convenience, all the characters are named as P0 to P4 – P0: John Watson, P1: Sherlock Holmes, P2: Stamford, P3: Jane Eyre, P4: Mr. Rochester.

As illustrated in Fig. 29., with dataset A, P3 achieves all the predictions 100 percent in every classifier, whereas P0 and P1 have less accurate predictions compared to P3. It is suspected that this is because P3 has less count of data but up-samples as the same amount of P0 and P1, which occurs the repetition of the similar data and let the machine easy to predict. However, Fig. 30 (a) shows the greatest accuracy achieving predictions of P0 and P1 about half.

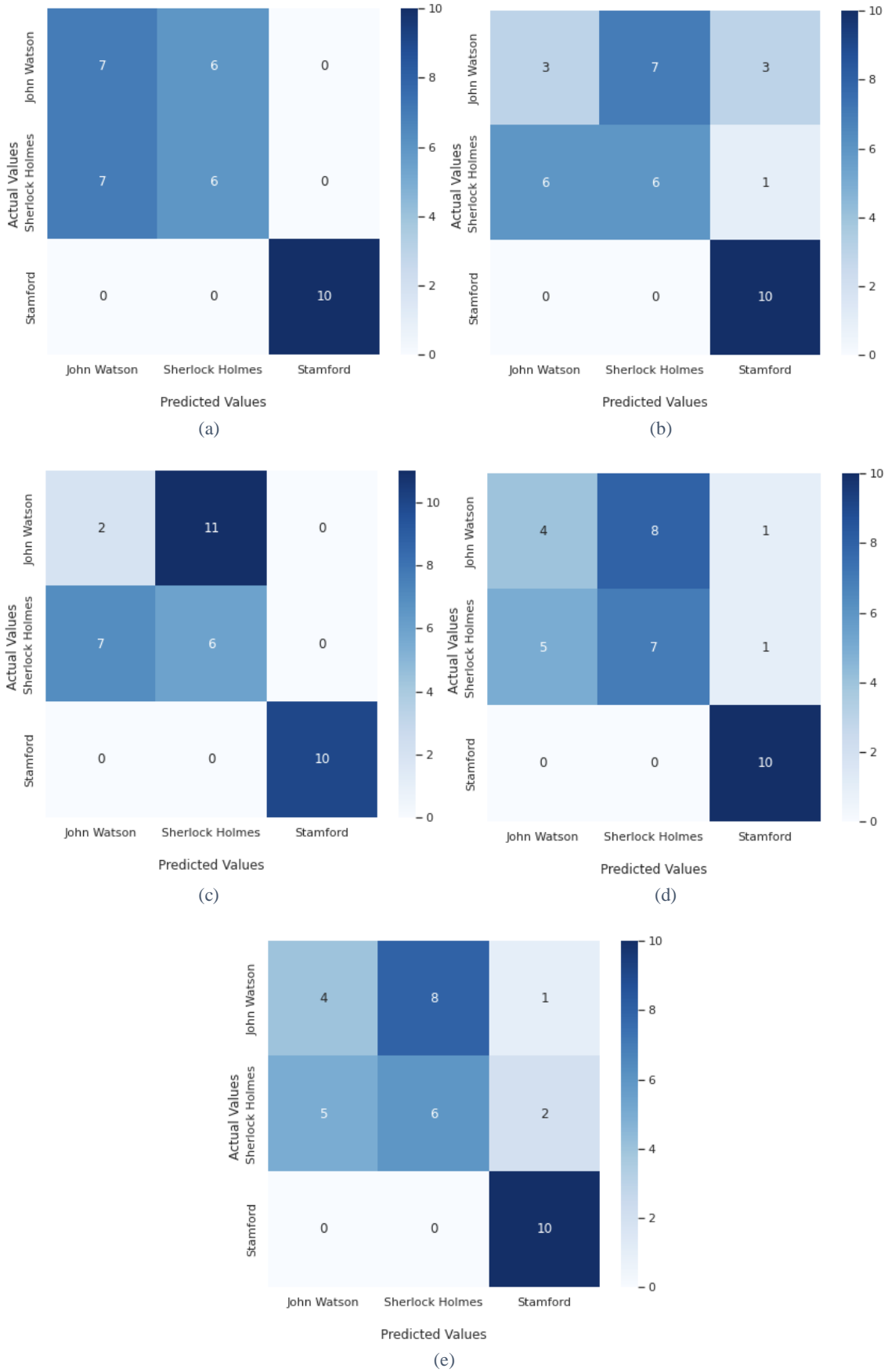


Fig. 29. Confusion matrix of dataset A. (a) RF. (b) GNB. (c) BNB. (d) MNB. (e) CNB.

However, dataset B, it shows more ideal confusion matrix in general compared to dataset A. Especially, Fig. 30 (d) and Fig. 30 (e) show the most accurate confusion matrix attaining about 80 percent accuracy for both P3 and P4. It is interesting that dataset B has shown higher accuracy compared to dataset A. It is suspected that the result might be influenced by the character's gender difference. Specifically, P0, P1 and P2, who are the characters from dataset A, have the same gender but P3 and P4, from dataset B, have the different gender. However, it is hard to conclude with only one comparison group so this can be investigated further with more datasets to judge.

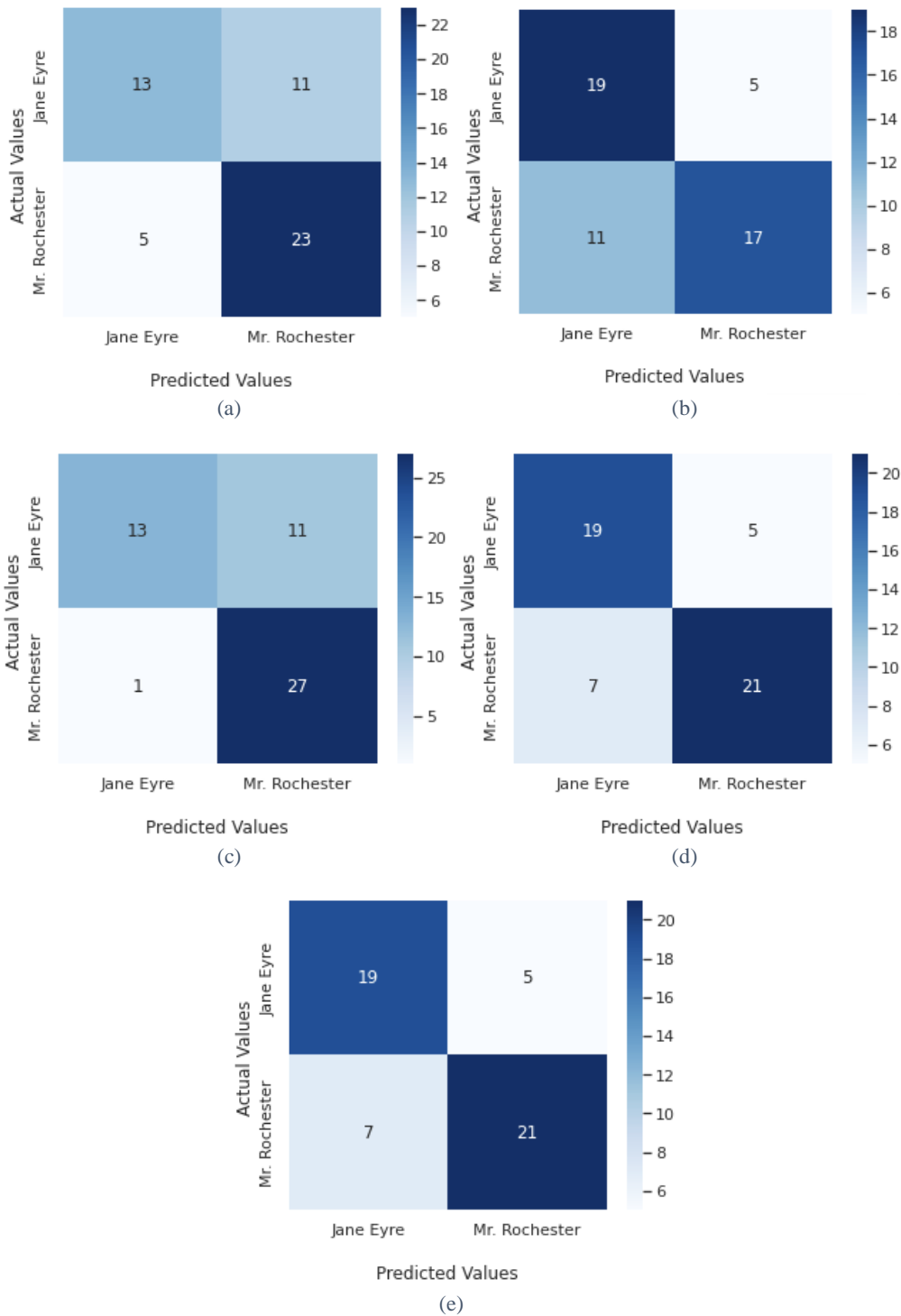


Fig. 30. Confusion matrix of dataset B. (a) RF. (b) GNB. (c) BNB. (d) MNB. (e) CNB.

5.2.4 Performance Metrics

The performance metrics consists of precision, recall and F1-score which compute like (14), (15) and (16).

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (14)$$

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (15)$$

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (16)$$

Comprehensively, in Fig. 31., Random Forest results in the highest accuracy in every score more than 60 percent with dataset A. In particular, recall shows the highest value of 67 among them. This seems to happen with other classifiers also as recall is the always highest. In addition, Multinomial NB and Complement NB are the second and third highest classifiers. The result values are visible in Appendix E.

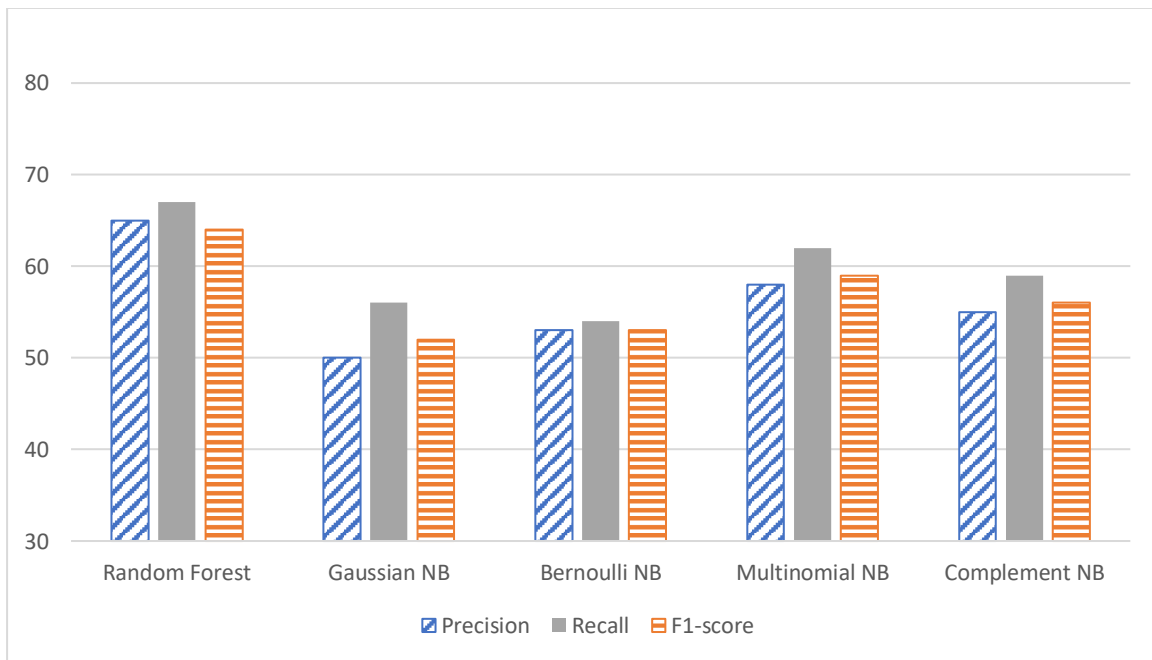


Fig. 31. Comparison of dataset A

On the other hand, as seen in Fig. 32., Random Forest still turns out with high accuracy, but other classifiers have derived a higher results than Random Forest, especially Bernoulli NB, Multinomial NB and Complement NB. Nevertheless, it is difficult to say that Random Forest is not accurate because all the classifiers including Random Forest result in high accuracy. The precise values are visible in Appendix E.

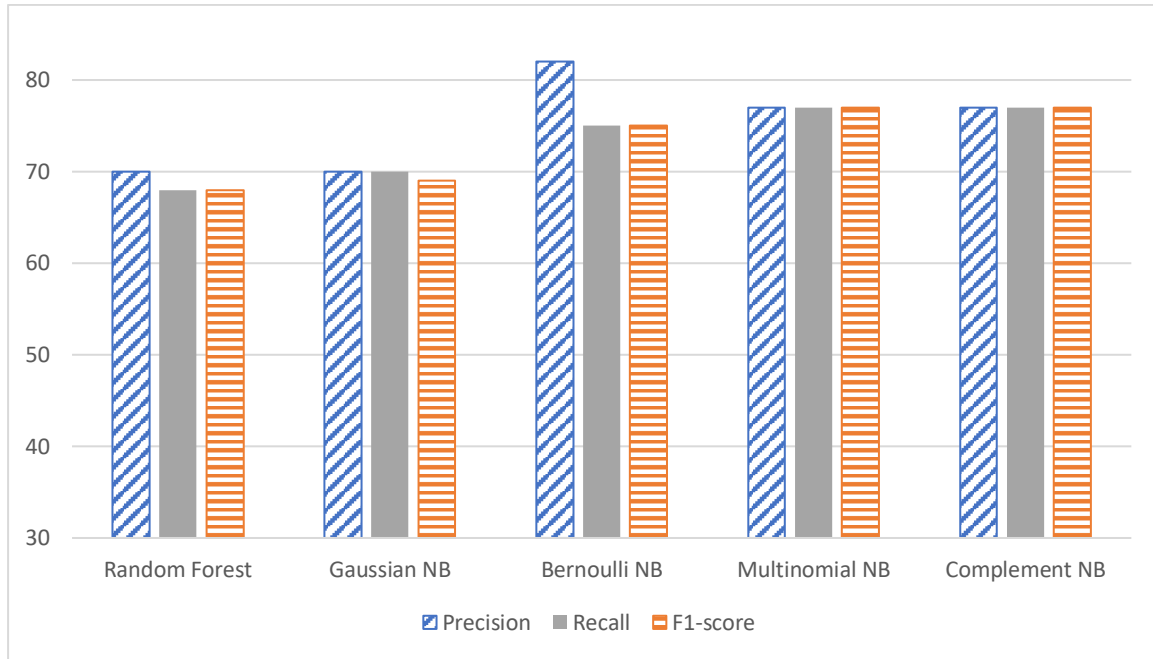


Fig. 32. Comparison of dataset B

5.3 User Interface

Finally, the result of the final machine learning model is tested based on the evaluation above. As a result, the machine is developed with Random Forest Classifier and trained to profile the character from literature. If the machine successfully guesses the characters from the input sentence, then it means profiling is succeeded.

As Fig. 33. shows, when a sentence is entered, then the speaker's name comes up. With this UI, it is tested with the dialogue from the same book but in another chapter which has not been recognised by AI model. Also, for the precise results, the test has done 10 times for each character – P0 to P4. However, as P3 has not been shown in other chapters, it has been removed from the test. The full test results are visible in Appendix F.

Character Profiling

Enter a sentence

All is changed about me, sir; I must change too—

Confirm

Speaker

Jane Eyre

Fig. 33. Guessing speaker's name by entered sentence

As Fig. 34. and Fig. 35. show, the result is quite accurate for both datasets. P0 and P1 get 7 right guessing out of 10, P3 for 8 and P4 for 9. Since every character achieves an average 77 percent accuracy, this project accomplishes its main purpose.

		Predicted Value	
Actual Value		P0	P1
	P0	7	3
	P1	3	7

Fig. 34. Confusion matrix of P0 and P1

		Predicted Value	
Actual Value		P3	P4
	P3	8	2
	P4	1	9

Fig. 35. Confusion matrix of P3 and P4

Summary

Interestingly, dataset A and B both obtain lower accuracy when applying coreference resolution due to the low performance of the method. As a result, the entire project and testing move on without implementing coreference resolution.

In general, dataset B has shown higher accuracy than dataset A. It is suspected that this is because of the gender difference between characters, as P0, P1 and P2 from dataset A all have the same gender and P3 and P4 from dataset B have different gender. The feature difference in gender may let the machine recognise the difference between characters better.

In addition, Bernoulli NB, Multinomial NB and Complement NB have shown higher performance with dataset B. However, it is hard to say these three are the best classifiers since it has lower accuracy with dataset A, especially Bernoulli NB has the lowest accuracy. However, Random Forest Classifier appears to have the highest accuracy with dataset A. Even if it has the lowest accuracy with dataset B, the value of precision, recall and F1-score are actually higher than what was derived with dataset A. Thus, in general, Random Forest has the highest performance among five different classifiers.

Moreover, although dataset A shows lower accuracy than dataset B, testing with the UI seems quite high showing 70 percent accuracy.

6. Conclusion

This chapter provides a conclusion which is based on the aim and objectives set at the beginning of this paper. It will deliver the overall summary of the initial purpose and the review of the goals, as well as propose a discussion of prospective further study.

6.1 Overall Summary

The overall aim of this dissertation was to develop an artificial intelligence (AI) model that profiles characters in the literature by analysing their dialogues and applying different approaches to improve its performance. To achieve this purpose, it is divided into a set of objectives. This section will assess how the goals were accomplished and what strategy was applied, and what could have been done better. At the last, it will summarise what have been learnt and suggests the further work.

6.1.1 Assessment of Aim and Objectives

- **Collect a set of datasets from raw text using Natural Language Processing (NLP) techniques.**

This objective has been fully met. The tool selected was SpaCy to implement NLP technology. It has been chosen based on the study result of comparison with other tools. Also, tokenisation, POS tagging, and dependency parsing method have been completed to extract the set of speaker and dialogue data from the literature document. However, the trial for applying Coreference Resolution was not successful when converting the pronoun into the speaker's name. Since Neuralcoref shows high accuracy for short sentences not long data, it is suggested to investigate different coreference tools such as Stanford CoreNLP and others.

- **Develop the model that can profile different kinds of literature so that results can be compared and analysed**

This goal has been satisfied. The model can extract two different datasets successfully and the distinction between the two results was analysed in depth by obtaining several graphs, the confusion matrix and performance score. However, it is based on the recent dialogue tag rule of English literature so it might not be compatible with other literature following other rules. Therefore, it could improve by adding several different parsers for another form of the book.

- **Implement several popular classifiers on text data.**

This requirement has been successfully fulfilled. Based on the research, five different classifiers from Scikit-learn were selected such as Random Forest, Gaussian Naïve Bayes, Bernoulli Naïve Bayes, Multinomial Naïve Bayes and Complement Naïve Bayes. Also, the result was compared to find the most promising one. Several evaluation methods were evaluated such as confusion matrix, F1-score, Recall and Precision. However, only the machine learning methods are applied due to the limitation of the time for the project, so it is suggested to apply deep learning models such as convolutional neural network (CNN) to achieve higher performance.

- **Identify best parameters for the model by implementing Hyper-parameter tuning**

This objective has been fully achieved. The development tool used was GridSearchCV which finds the best parameters in the range of the value set. It improves the accuracy of all the classifiers compared to the performance before

applying GridSearchCV. Nonetheless, GridSearchCV can be slow if the model has many parameters, so it is recommended to try random search as a hyper-parameter tuning tool for better implementation with reducing complexity.

- **Evaluating user interface (UI) based on the web application**

This has been partially achieved. the design of UI have been successfully implemented on the web but not a web application since the framework used, Google Colab, is not compatible with the web application tool. However, using the Ipywidget library helps to execute UI on the Google Colab to display the final result. UI shows the entering space for a random sentence, and once it enters, the model guesses the character and shows the name of the speaker who likely said the sentence. To assess the accuracy of the result, manual checking has done ten times for each character and the accuracy turns out 77 percent on average. Nevertheless, in the future, it is proposed to try other web application tools such as React Native to implement the whole model to show on the UI.

Summary

Overall, the project went well satisfying its goals and aim. The technique of collecting dataset has successfully built and the hyper-parameter tuning with the training result shows state-of-the-art accuracy. In addition, the identification of the best classifiers among the five different method is also analysed clearly with its evidence.

6.1.2 Reflection

By attempting to meet the project's purpose, knowledge about how to process natural language to train them is gained. Also, the research of various libraries and its principles help to understand principle and the inner algorithms of the methods such as SpaCy and Scikit-learn's classifiers. Also, it provides the knowledge to determine which algorithm would be the most appropriate for a certain classification problem.

Moreover, due to the limitation of given time, there are several experiments have been remained as they are low priority of this project. For example, to identify the best hyper-parameter tuning tool, the various tools could have been compared and analysed to discover its differences and the reason of the exceptional results. Also, different coreference tools could have applied to succeed converting pronouns automatically and deep learning methods could have been applied to achieve better performance for the result. Finally, building web application with web framework could have done to show the entire process.

6.2 Future Work

Based on the natural language processing (NLP) techniques and character profiling, this project's capabilities can be extended in several areas such as expanding to various languages and further developing conversational AI.

This project was mainly focusing on character profiling from English literature. However, since the NLP tools, SpaCy and NLTK, support more than 10 languages, the project would be expanded to develop a model that is compatible with literature written in other languages. Analysing the results from different languages might give more interesting studies.

Moreover, based on the character profile created by this project, it can be expanded to conversational AI which has the character's personality. Even though the AI chatbot has been

utilised in various areas such as healthcare, education and services, it still has a limitation, a lack of human-like attributes. Thus, the chatbot based on the literature characters' personalities will help to overcome this drawback.

References

- [1] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261-266, 2015.
- [2] M. Kanakaraj and R. M. R. Guddeti, "NLP based sentiment analysis on Twitter data using ensemble classifiers," *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, pp. 1-5, 2015.
- [3] Y. Sang, X. Mou, M. Yu, S. Yao, J. Li and J. Stanton, "TVShowGuess: Character Comprehension in Stories as Speaker Guessing," 2022.
- [4] G. H. Bower and D. G. Morrow, "Mental models in narrative comprehension," *Science*, vol. 247, no. 4938, pp. 44-48, 1990.
- [5] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela and J. Weston, "Personalizing Dialogue Agents: I have a dog, do you have pets too?," *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 2204-2213, 2018.
- [6] S. Rathi, J. P. Verma, R. Jain, A. Nayyar and N. Thakur, "Psychometric profiling of individuals using Twitter profiles: A psychological Natural Language Processing based approach," 2022.
- [7] M. Haenlenin and A. Kaplan, "A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence," *California Management Review*, vol. 61, no. 4, pp. 5-14, 2019.
- [8] F. Raso, H. Hilligoss, V. Krishnamurthy, C. Bavitz and L. Kim, "Artificial Intelligence & Human Rights: Opportunities&Risks," *Berkman Klein Center for Internet & Society Research Publication*, p. 63, 2018.
- [9] X. Du-Harpur, F. M. Watt, N. M. Luscombe and M. D. Lynch, "What is AI? Applications of artificial intelligence to dermatology," *British Journal of Dermatology*, vol. 183, no. 3, pp. 423-430, 2020.
- [10] J. Hurwitz and D. Kirsch, *Machine Learning for Dummies*, IBM Limited Edition, Hoboken: John Wiley & Sons, Inc, 2018.
- [11] I. E. Naqa and M. J. Murphy, *Machine Learning in Radiation Oncology*, 2015.
- [12] D. Sarkar, *Text Analytics with Python: A Practitioner's Guide to Natural Language Processing*, Apress, 2019.
- [13] "Natural Language Processing (NLP)," IBM, 2 July 2020. [Online]. Available: <https://www.ibm.com/cloud/learn/natural-language-processing>. [Accessed 4 Mar 2022].
- [14] F. N. A. A. Omran and C. Treude, "Choosing an NLP Library for Analyzing Software Documentation: A Systematic Literature Review and a Series of Experiments," *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pp. 187-197, 2017.
- [15] "Tokenization," SpaCy, 2022. [Online]. Available: <https://spacy.io/usage/linguistic-features#how-tokenizer-works>. [Accessed 4 May 2022].
- [16] A. Voutilainen, "Part-of-Speech Tagging," in *The Oxford Handbook of Computational Linguistics*, New York, Oxford University Press Inc., 2003, pp. 219-230.
- [17] S. Tyagi and G. S. Mishra, "Statistical Analysis of Part of Speech (POS) Tagging Algorithms for English Corpus," *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 2, no. 3, pp. 1-9, 2016.

- [18] C. P. Harshitha and N. R. Sunitha, "Topic Identification for Semantic Grouping based on Hidden Markov Model," *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pp. 932-937, 2020.
- [19] A. S. Shafie, N. M. Sharef, M. A. A. Murad and A. Azman, "Aspect Extraction Performance with POS Tag Pattern of Dependency Relation in Aspect-based Sentiment Analysis," *2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP)*, pp. 1-6, 2018.
- [20] S. Kübler, R. McDonald and J. Niver, *Synthesis Lectures on Human Language Technologies*, Morgan & Claypool Publishers, 2009.
- [21] H. Christian, M. P. Agus and D. Suhartono, "Single Document Automatic Text Summarization using Term Frequency-Inverse Document Frequency (TF-IDF)," *ComTech Computer Mathematics and Engineering Applications*, vol. 7, no. 4, 2016.
- [22] F. Amato, L. Coppolino, G. Cozzolino, G. Mazzeo, F. Moscato and R. Nardone, "Enhancing random forest classification with NLP in DAMEH: A system for Data Management in eHealth Domain," *Neurocomputing*, vol. 444, pp. 79-91, 2021.
- [23] V. Jackins, S. Vimal, M. Kaliappan and M. Y. Lee, "AI-based smart prediction of clinical disease using random forest classifier and Naive Bayes," *Journal of Supercomputing*, vol. 77, no. 5, pp. 5198-5219, 2020.
- [24] A. H. Jahromi and M. Taheri, "A non-parametric mixture of Gaussian naive Bayes classifiers based on local independent features," *2017 Artificial Intelligence and Signal Processing Conference (AISP)*, pp. 209-212, 2017.
- [25] G. Singh, B. Kumar, L. Gaur and A. Tyagi, "Comparison between Multinomial and Bernoulli Naïve Bayes for Text Classification," *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, pp. 593-596, 2019.
- [26] J. D. M. Rennie, L. Shih, J. Teevan and D. R. Karger, "Tackling the Poor Assumptions of Naive Bayes Text Classifiers," *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, 2003.
- [27] L. Pang, "Speaker Extraction," GitLab, [Online]. Available: <https://gitlab.com/snowhitiger/speakerextraction>. [Accessed 1 May 2022].
- [28] "What is Python? Executive Summary," Python, 2022. [Online]. Available: <https://www.python.org/doc/essays/blurb/>. [Accessed 10 May 2022].
- [29] "Colaboratory," Google, 2022. [Online]. Available: <https://research.google.com/colaboratory/faq.html>. [Accessed 10 May 2022].
- [30] S. Bird, E. Klein and E. Loper, *Natural Language Processing with Python*, O'Reilly Media, Inc., 2009.
- [31] M. Wang and F. Hu, "The Application of NLTK Library for Python Natural Language Processing in Corpus Research," *Theory and Practice in Language Studies*, vol. 11, no. 9, pp. 1041-1049, 2021.
- [32] D. Sarkar, R. Bali and T. Sharma, *Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems*, Apress, 2017.
- [33] Y. Vasiliev, *Natural Language Processing with Python and spaCy: A Practical Introduction*, No Starch Press, 2020.
- [34] X. Schmitt, S. Kübler, J. Robert, M. Papadakis and Y. LeTraon, "A Replicable Comparison Study of NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate,"

2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), pp. 338-343, 2019.

- [35] “How to Write Dialogue: 9 Practical Tips for Writers (+ Examples),” reedsy, 1 Dec 2021. [Online]. Available: https://blog.reedsy.com/guide/how-to-write-dialogue/#2__keep_to_three_dialogue_beats. [Accessed 7 May 2022].
- [36] Huggingface, “NeuralCoref 4.0: Coreference Resolution in spaCy with Neural Networks,” Github, 2021. [Online]. Available: <https://github.com/huggingface/neuralcoref>. [Accessed 8 May 2022].
- [37] A. Patel and K. Meehan, “Fake News Detection on Reddit Utilising CountVectorizer and Term Frequency-Inverse Document Frequency with Logistic Regression, MultinomialNB and Support Vector Machine,” *2021 32nd Irish Signals and Systems Conference (ISSC)*, pp. 1-6, 2021.
- [38] N. A. S. Nilam, F. M. A. Nurulhuda, C. Suriyati, M. S. Haslina, Y. Yazriwati and M. S. Suriani, “SMS Spam Message Detection using Term Frequency-Inverse Document Frequency and Random Forest Algorithm,” *Procedia Computer Science*, vol. 161, no. 1877-0509, pp. 509-515, 2019.
- [39] A. C. Müller and S. Guido, *Introduction to Machine Learning with Python*, O'Reilly Media, Inc..
- [40] Y. Shuai, Y. Zheng and H. Huang, “Hybrid Software Obsolescence Evaluation Model Based on PCA-SVM-GridSearchCV,” *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, pp. 449-453, 2018.
- [41] “ipywidgets,” Jupyter widgets, 10 May 2022. [Online]. Available: <https://ipywidgets.readthedocs.io/en/latest/>. [Accessed 11 May 2022].
- [42] “Dialogue Examples: 15 Great Passages of Dialogue, Analyzed,” Reedsy, 14 Jan 2021. [Online]. Available: <https://blog.reedsy.com/guide/how-to-write-dialogue/dialogue-examples/>. [Accessed 1 May 2022].
- [43] T. Yiu, “Understanding Random Forest,” Medium, 12 Jun 2019. [Online]. Available: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>. [Accessed 12 May 2022].
- [44] “Naive Bayes,” Scikit-learn, 2022. [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html. [Accessed 12 May 2022].
- [45] tartila, “Agile,” Freepik, [Online]. Available: https://www.freepik.com/premium-vector/agile-development-process-infographic-software-developers-sprints-product-management-scrum-sprint-scheme-illustration_8636959.htm#query=agile&position=10&from_view=search. [Accessed 13 May 2022].

Appendices

Appendix A

F1-score result with coreference resolution and without it

Dataset A

Dataset A	original	coreference resolution
Random Forest	61	26
Gaussian NB	47	11
Bernoulli NB	58	31
Multinomial NB	55	31
Complement NB	56	5

Dataset B

Dataset B	original	coreference resolution
Random Forest	73	43
Gaussian NB	69	36
Bernoulli NB	65	43
Multinomial NB	59	47
Complement NB	58	23

Appendix B

F1-score result with hyper-parameter tuning and without it

Dataset A

Dataset A	Before	After
Random Forest	61	64
Gaussian NB	47	53
Bernoulli NB	58	50
Multinomial NB	55	58
Complement NB	56	56

Dataset B

Dataset B	Before	After
Random Forest	73	69
Gaussian NB	69	69
Bernoulli NB	65	77
Multinomial NB	59	77
Complement NB	58	77

Appendix C

F1-score result of Dataset A, Dataset B and Average

F1-score	Dataset A	Dataset B	Average
Random Forest	64	69	66.5
Gaussian NB	53	69	61
Bernoulli NB	50	77	63.5
Multinomial NB	58	77	67.5
Complement NB	56	77	66.5

Appendix D

Cross-validation result of Dataset A, Dataset B and Average

Cross-validation	Dataset A	Dataset B	Average
Random Forest	73	77	75
Gaussian NB	72	74	73
Bernoulli NB	78	70	74
Multinomial NB	72	79	75.5
Complement NB	71	78	74.5

Appendix E

Result of Precision, Recall, F1-score

Dataset A

Dataset A	Precision	Recall	F1-score
Random Forest	65	67	64
Gaussian NB	50	56	52
Bernoulli NB	53	54	53
Multinomial NB	58	62	59
Complement NB	55	59	56

Dataset B

Dataset B	Precision	Recall	F1-score
Random Forest	70	68	68
Gaussian NB	70	70	69
Bernoulli NB	82	75	75
Multinomial NB	77	77	77
Complement NB	77	77	77

Appendix F

Result of Speaker Guessing

Dataset A

Input sentence	Actual speaker	Predicted speaker	Result
You amaze me, Holmes	John Watson	Sherlock Holmes	Incorrect
Surely you are not as sure as you pretend to be of all those particulars which you gave.	John Watson	Sherlock Holmes	Incorrect
That seems simple enough	John Watson	Sherlock Holmes	Incorrect
but how about the other man's height?	John Watson	John Watson	Correct
And his age?	John Watson	John Watson	Correct
What ineffable twaddle!	John Watson	John Watson	Correct
I never read such rubbish in my life.	John Watson	John Watson	Correct
Why, this article	John Watson	John Watson	Correct
And how?	John Watson	John Watson	Correct
And these other people?	John Watson	John Watson	Correct
There's no room for a mistake	Sherlock Holmes	Sherlock Holmes	Correct
The very first thing which I observed on arriving there was that a cab had made two ruts with its wheels close to the curb. Now, up to last night, we have had no rain for a week, so that those wheels which left such a deep impression must have been there during the night. There were the marks of the horse's hoofs, too, the outline of one of which was far more clearly cut than that of the other three, showing that that was a new shoe. Since the cab was there after the rain began, and was not there at any time during the morning--I have Gregson's word for that--it follows that it must have been there during the night, and, therefore, that it brought those two individuals to the house.	Sherlock Holmes	Sherlock Holmes	Correct
Why, the height of a man, in nine cases out of ten, can be told from the length of his stride. It is a simple calculation enough, though there is no use my boring you with figures. I had this fellow's stride both on the clay outside and on the dust within. Then I had a way of checking my calculation. When a man writes on a wall, his instinct leads him to write about the level of his own eyes. Now that	Sherlock Holmes	John Watson	Incorrect

writing was just over six feet from the ground. It was child's play.			
What is it?	Sherlock Holmes	Sherlock Holmes	Correct
I see that you have read it since you have marked it. I don't deny that it is smartly written. It irritates me though. It is evidently the theory of some arm-chair lounge who evolves all these neat little paradoxes in the seclusion of his own study. It is not practical. I should like to see him clapped down in a third class carriage on the Underground, and asked to give the trades of all his fellow-travellers. I would lay a thousand to one against him.	Sherlock Holmes	Sherlock Holmes	Correct
You would lose your money	Sherlock Holmes	Sherlock Holmes	Correct
As for the article I wrote it myself.	Sherlock Holmes	Sherlock Holmes	Correct
Yes, I have a turn both for observation and for deduction. The theories which I have expressed there, and which appear to you to be so chimerical are really extremely practical--so practical that I depend upon them for my bread and cheese.	Sherlock Holmes	Sherlock Holmes	Correct
Well, I have a trade of my own. I suppose I am the only one in the world. I'm a consulting detective, if you can understand what that is. Here in London we have lots of Government detectives and lots of private ones. When these fellows are at fault they come to me, and I manage to put them on the right scent. They lay all the evidence before me, and I am generally able, by the help of my knowledge of the history of crime, to set them straight. There is a strong family resemblance about misdeeds, and if you have all the details of a thousand at your finger ends, it is odd if you can't unravel the thousand and first. Lestrade is a well-known detective. He got himself into a fog recently over a forgery case, and that was what brought him here.	Sherlock Holmes	John Watson	Incorrect

They are mostly sent on by private inquiry agencies. They are all people who are in trouble about something, and want a little enlightening. I listen to their story, they listen to my comments, and then I pocket my fee.	Sherlock Holmes	John Watson	Incorrect
---	-----------------	-------------	-----------

Dataset B

Input sentence	Actual speaker	Predicted speaker	Result
That I am not Edward Rochester's bride is the least part of my woe	Jane Eyre	Jane Eyre	Correct
that I have wakened out of most glorious dreams, and found them all void and vain, is a horror I could bear and master; but that I must leave him decidedly, instantly, entirely, is intolerable. I cannot do it.	Jane Eyre	Jane Eyre	Correct
Let me be torn away, then	Jane Eyre	Jane Eyre	Correct
Let another help me!	Jane Eyre	Jane Eyre	Correct
I cannot: I am tired and sick. I want some water.	Jane Eyre	Mr. Rochester	Incorrect
Much better, sir; I shall be well soon.	Jane Eyre	Jane Eyre	Correct
Yes, sir.	Jane Eyre	Mr. Rochester	Incorrect
At any rate, there is neither room nor claim for me, sir.	Jane Eyre	Jane Eyre	Correct
Sir, I do not wish to act against you	Jane Eyre	Jane Eyre	Correct
All is changed about me, sir; I must change too—there is no doubt of that; and to avoid fluctuations of feeling, and continual combats with recollections and associations, there is only one way—Adèle must have a new governess, sir.	Jane Eyre	Jane Eyre	Correct
Jane!	Mr. Rochester	Mr. Rochester	Correct
Lingerer!	Mr. Rochester	Mr. Rochester	Correct
my brain is on fire with impatience, and you tarry so long!	Mr. Rochester	Mr. Rochester	Correct
Is John getting the carriage ready?	Mr. Rochester	Mr. Rochester	Correct
Is the luggage brought down?	Mr. Rochester	Mr. Rochester	Correct
Go you to the church: see if Mr. Wood (the clergyman) and the clerk	Mr. Rochester	Mr. Rochester	Correct

are there: return and tell me.			
And the carriage?	Mr. Rochester	Mr. Rochester	Correct
We shall not want it to go to church; but it must be ready the moment we return: all the boxes and luggage arranged and strapped on, and the coachman in his seat.	Mr. Rochester	Jane Eyre	Incorrect
Jane, are you ready?	Mr. Rochester	Mr. Rochester	Correct
Am I cruel in my love?	Mr. Rochester	Mr. Rochester	Correct