Authorship Attribution with Markov Chain

Teng Li Yuan
201387914

Supervised by Professor Charles Taylor, Dr. Jochen Voss and Dr. Matthew Aldridge

Submitted in accordance with the requirements for the
module MATH5871M: Dissertation in Statistics
as part of the degree of

Master of Science Data Science and Analytics

The University of Leeds, School of Mathematics

October 2020

The candidate confirms that the work submitted is his/her own and that appropriate credit has been given where reference has been made to the work of others.

Abstract

Given a piece of work, how can we identify the author? Authorship attribution is the task of identifying the author of a given text. The idea behind Markov chain is the future state depends on current state but independent of its past. With the assumption of every individual has a unique and consistent writing style, Markov chain can be applied as the transition probability of the next character given current word varies between everyone.

The project was implemented following the Cross Industry Industry Standard Process for Data Mining (CRISP-DM) framework. A collection of 585 ebooks from Project Gutenberg were used as the data set to carry out the study. The written sample of the candidates and the unknown text were tokenised into sequences of character and subsequently constructed into the transition matrices. Four methods including Log-likelihood, Kullback–Leibler divergence, Jensen-Shannon Divergence and Linear Discriminant analysis were proposed for classification task. Next, the classification model with higher order Markov chain were implemented. Lastly, ensemble model with majority voting integrates all the predictions from its member model and output a single prediction.

Higher order Markov chain with more transition probabilities allow more stylometry features to be used in authorship attribution. Thus, its result shows much better model performance than first order Markov chain. In this study, third order Markov chain Kullback–Leibler divergence model is the best performing model and produce a discriminant result. However, ensemble model that uses various method at different Markov chain order was recommended for its balance class prediction and reliability performance. The work also presented some workaround to mitigate negative impact of zero-probability event.

Acknowledgement

I would like to express my gratitude towards my project supervisors, Professor Charles Taylor, Dr. Jochen Voss and Dr. Matthew Aldridge for their guidance and support in the past three months, especially in the light of Covid-19 pandemic. Lastly, I would like to thank the University of Michigan for the sharing of the pre-cleaned Gutenberg data set.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ever since the existence of words, words are constantly written by human to convey messages, expression of feeling and documentation. While words were delivered to the reader, the interest of author's identity sometimes exceed the content itself. Identity of the author influence the reader's attention to message. For instance, the same sentence appeared on Donald Trump's twitter clearly attracts more attention than the same sentence written by a normal person.

Authorship attribution is the task of identifying the author of a given text from a group of candidates based on the writing samples (Stamatatos 2009). The main idea behind the task is that by extracting some measurable textual features, the true author can be identified with similar features. It is always confused with author profiling that study the author's information such as age, gender and personality (Argamon et al. 2003). Authorship attribution has a broad application in areas such as fraud detection (Suman Patil 2019), computer forensic (Tennyson 2012, Yang et al. 2017), plagiarism detection (Ramnial et al. 2016), literature research (Hoover 2004) and cyber crime detection (Rastogi & Shivam 2018). Due to the proliferation of social media, the recent study of authorship attribution has shifted to social media forensic (Rocha et al. 2017).

Thomas C. Mendenhall (Mandenhal 1887) is one of the first pioneer attempted to characterise authors by the frequency distribution of words with various lengths. This traditional human expert-based methods has attracted much interest of the public and subsequently been applied to answer for the controversial Shakespeare authorship question.

Authorship attribution also play an important role in historical forensic. The Federalist Papers dispute is perhaps one of the most well-known study case for authorship attribution. The Federalist paper consists of series of published article during year 1787 to 1788 where authorship of 12 articles were claimed by two different authors. However, the debate remain unsolved until almost three century later Mosteller & Wallace (1964) proposed a Bayesian statistic analysis based on the frequency of the common words that explicitly discriminate between the two authors. The study catalysed the research of authorship attribution using non-traditional authorship attribution.

The proposed methods during that time were computer-assisted rather than computer-based

(Stamatatos 2009), where most of the study are not focusing to develop a fully-automated model. It is not until the late 1990s where the plethora of electronic texts in the internet that the development of fully-automated authorship attribution task rapidly developed. Stamatatos (2009) had summarised the main stylometry features used in authorship attribution including:

1. Lexical features where the text can be tokenised into tokens of word, punctuation mark and number. Corresponding lexical features are word length, sentence length and word frequencies.

2. Characters features where text can be tokenised into tokens of character. Corresponding character features are letter frequencies, character n-grams, character counts and etc.

3. Syntactic features that make uses of syntactic information unconsciously by the authors. This includes errors, rewrite rules frequencies, sentence and phrase structure.

4. Semantic features that make uses of syntactic information unconsciously by the authors. This includes synonyms, functional, sentence and semantic dependencies.

5. Application specific features that can only be applied on specific platform or domain. For instance, use of greetings in the emails, use of indentation, font color count and font size count in HTML form.

Markov chain (MC) is a stochastic process that model probability of future state solely by its current state. It was introduced by the Russian mathematician Andrey Markov. As texts can be viewed as sequence of words or characters, where the next word or character is governed by stylometric style of author, Markov chain can be applied and the transition probabilities between words can be used as an authorship fingerprint to distinguish between authors.

The study will be conducted according to the Cross Industry Industry Standard Process for Data Mining (CRISP-DM) framework while present the finding in standard report manner. This report were organised in the following way; section 1 introduced the background, objectives, and CRISP-DM framework. Subsequently, section 2 focus on the understanding and preparation of data set. Various data quality issues, characters features selection and cleaning process were discussed in this section. Next, section 3 discussed the project methodology includes Markov chain, similarity distance measurement and the detailed implementation of the model. Section 4 evaluate the various models' performance at different order Markov Chain. Lastly, conclusions were made in section 5.

1.1 CRISP-DM

Data mining is the process of discovering interesting pattern and knowledge from large amount of data (Han et al. 2011). In fact, authorship attribution task in this project is a data mining process that aimed to classify the authorship of the given text to by measuring its stylometry features with the training data. Data mining is a relatively young field that emerge and develop rapidly after explosive growth of data. For this reason, Cross Industry Standard Process for Data Mining (CRISP-DM), a framework for data mining process was introduced under the ESPRIT program funded by European Commission(Shearer 2000). CRISP-DM contains six phases that described the life cycle of a data mining project.

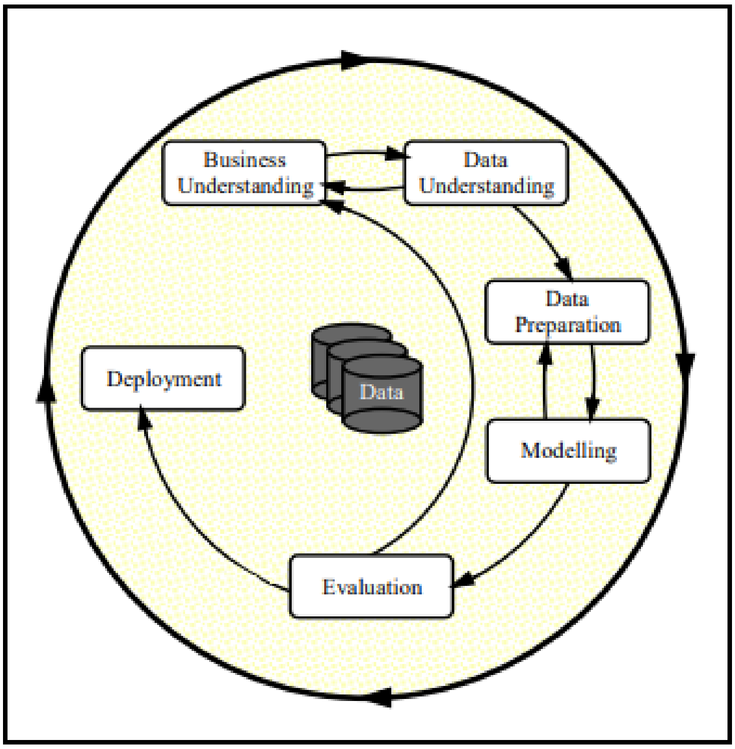


Figure 1.1: Six phases of CRISP-DM

Figure 1.1 illustrates the six phases of CRISP-DM process. *Business understanding* defines the objectives, requirements, success criteria and goals. The next phase is *data understanding* that focus on data collection, data exploration and data quality verifying. After getting insight of the data, *data preparation* involves selection, cleaning, integration and reformatting of data as well as the preparation of the data set for the project. *Modeling* phase determines the modeling technique, parameters setting and algorithms. The fifth phase of CRISP-DM is *evaluation* which focus on evaluation of the performance and review with objective and successful criteria set. The final step is *deployment* where involves planning for deployment, monitoring and maintenance. In addition, it also include documentation of final report and project review.

Some phases in CRISP-DM are reversible. For example, the user can revert back from modeling to data preparation phase if the thinks the data set requires additional cleaning and features extraction.

CRISP-DM provides a generic guidance for any data mining project of any field. This framework is applied as the core methodology in this project. Business understanding is discussed in chapter 1, data understanding and preparation is presented in chapter 2. Chapter 3 resem ble the modelling phase and the evaluation is discussed in chapter 4.

1.2 Project Objectives

There are three main objectives that aimed to be achieved at the end of the project.

1. Developed an authorship attribution model with at least 80% accuracy and F1 score of 0.8 in 20 authors using Markov chain.

2. Implementation of CRISP-DM in the development process

3. Investigate the model performance in higher order Markov chain

Chapter 2

Data Understanding and Data Preparation

Good data understanding and data preparation built the solid foundation of data mining. This chapter starts by introducing the project data set and the characters distribution analysis. Data quality issues in the data set were highlighted with examples. Next, necessary data cleaning to address the data quality issues were executed and the data set were split into non-overlapping training set and testing set.

2.1 Exploratory Data Analysis

2.1.1 Data Set

Project Gutenberg (PG) is an online digital library that aims to encourage the creation and distribution of eBooks (*Project Gutenberg* 2018). It is recognised as the oldest digital library in the world. Project Gutenberg was founded by American writer Michael Stern Hart in the year 1971, long before the World Wide Web publicly available in 1991.

Today, Project Gutenberg has more than 60,000 free eBooks in its collection (*Project Gutenberg* 2018). Most of the published eBooks in the collection had received copyright clearance, making it an ideal testing ground for research.

Gutenberg data set is a small subset from PG corpus prepared by University Michigan. It consists of 3036 English books written by 142 authors (Lahiri 2014). Due to project timeline and computation power restriction, a small subset of Gutenberg data was selected to be the project data set. The *project data set* consists of 585 eBooks from 20 authors who contributes at least 5 English books. This selection criteria is to ensure the model has adequate data to be trained and learn the underlying relationship.

2.1.2 Characters Distribution Analysis

As this study focus on using character-level features in authorship attribution, character distribution analysis is essential for the feature selection and data cleaning. There are total of 244,422,749 characters in the data set, made by 165 different characters from the collection of 585 eBooks. The characters can be categorised into three main categories: **Alphabet**, **Numeric** and **Symbol**.

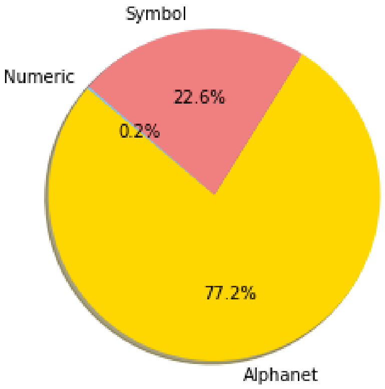


Figure 2.1: Character distribution in project data set

Figure 2.1 shows the character distribution in the project data set. Alphabet characters dominates the data set with 77.2% of the text while symbols contributes 22.6% of the text. On the other hand, numeric characters are the minority which occupied only 0.2% of the text. Thus, the analysis will be focused only on the characters and symbols.

Character	Percentage / %
blank space	16.84
e	9.62
t	7.09
a	6.26
o	5.97
n	5.25
h	5.20
i	5.14
s	4.86
r	4.52

Table 2.1: Highest character frequency (alphabets, symbols and numbers) in the data set

Table 2.1 shows the 10 highest frequency character in the data set. As expected, the blank space character (“ ”) is the highest frequency characters in the data set. Interestingly, despite having 165 characters in the data set, the top ten characters “ etaonhisr” made up of 70.75% of text in the data set.

Looking only at alphabet characters, 99.9% of the alphabets characters belongs to the 26

Alphabet	Percentage / %
e	12.45
t	9.18
a	8.11
o	7.73
n	6.80

Table 2.2: Highest frequency alphabet characters in the data set

Alphabet	Percentage / %
k	0.70
x	0.15
j	0.13
q	0.11
z	0.05

Table 2.3: Lowest frequency alphabet characters in the data set

English characters (A-Z). Five highest frequency alphabets descending order are “etaon” as shown in table 2.2. The alphabet distribution of the project data set is similar with Samuel Morse’s study. Samuel Morse, the inventor of Morse code studied letter frequency with the intention to assign simplest code to highest frequency characters. The letters “etain” were found to be the highest frequency (descending order) (Richardson et al. 2004). This is exactly the reason “e” and “t” are represent by single dot and single dash in Morse code. The lowest frequency English alphabets in ascending order order are “zqjxk”, occupying 1.14% of the total alphabets in project data set. There is also a negligible portion of the text written in Non-English characters like “æ” and “é”. These minority alphabet characters are approximately 0.01% of the total alphabets.

Symbol	Percentage / %
blank space	74.57
\n	8.79
,	6.05
.	4.12
”	1.90
’	1.35
-	1.31
Others	1.91

Table 2.4: Distribution of symbol category

Table 2.4 illustrates the distribution of symbol characters in the project data set. Symbols are characters that were neither alphabet or number, this includes punctuation mark, currency symbol (£), blank space and newline character (\n). It is found that 98.08% of the symbols are made up by the top 7 symbols while other symbols made up 1.91% of the total symbol characters.

2.2 Data Quality Issues

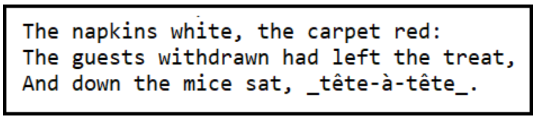
The consensus definition for data quality is “fitness for use” and subjective to judgement of data user (Wang & Strong 1996). Data quality issues are matter that lower that data quality. In authorship attribution, stylometry features of an author are extracted from the written sample. Original content of the author such as preface, content, acknowledgement are retain because it contained the stylometry features of that author.

On contrary, it is essential to exclude the texts which are not written by the author, as much as possible. Failing to remove these noises can result the model to learn inaccurate features. These includes metadata, license information and contributor notes that were present in the original eBook. Nevertheless, these noises in the project data set is extremely difficult to be cleaned by automation due to its inconsistency pattern.

Fortunately, Gutenberg data set was already manually pre-cleaned to remove most of the noise. Although additional manual cleaning is performed to further improve the data quality, the preprocessing of Gutenberg data set allows more effort to be put in modelling and analysis in lieu of data cleaning. Once again, we would like to express our gratitude to the contributors for sharing the data set to public.

2.2.1 Data Quality Issue - Minority characters

In section 2.1.2, minority characters that has low frequency in the data set were identified. This includes any non-English alphabets, numbers and minority symbols. In Markov chain analysis, probabilities of the character sequences were used as features in authorship attribution. These minority characters has a very low frequency, sometimes occurred only a few times in the data set can be a data quality issue. As a result, authorship attribution model can disqualifies the authorship of a candidate without consider other parts of the text. Zero probabilities event is undesirable but inevitable, the consequences of will be further discussed in chapter 3.

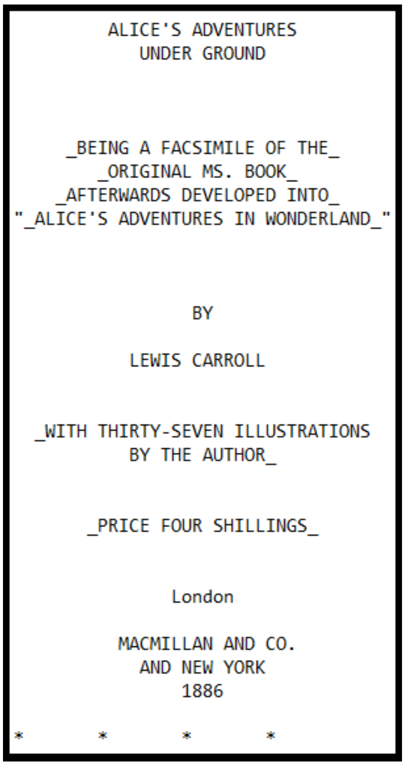


```
The napkins white, the carpet red:
The guests withdrawn had left the treat,
And down the mice sat, _tête-à-tête_.
```

Figure 2.2: Non-English characters used in the English literature - “The Poetical Works of Alexander Pope”

There are total of 10,353 non-English alphabets used in the data set. For example, in the book *“The Poetical Works of Alexander Pope”*, the word “tête-à-tête” originated from French is an adverb used to describe two person sit together in private. Very often, non-English alphabets are found in English literature. *Loanwords* are words borrowed from other languages with little or no modification (Kay 1995). Studies show that majority of the modern English words are derived from foreign language such as French (28.3%), Latin (28.24)% and German (25%) (Finkenstaedt & Wolff 1973). In fact, used of the loanwords are commonly seen even in country with majority native English speaker like UK. For instance, the word “café” and “Déjà vu” are originated from French. Although these loanwords can be spelled using English characters (“cafe” and “Deja vu”), some users still prefer its presence to enrich the flavor of their writing.

2.2.2 Data Quality issue - Consecutive blank spaces, newline characters, separation lines and line numbering

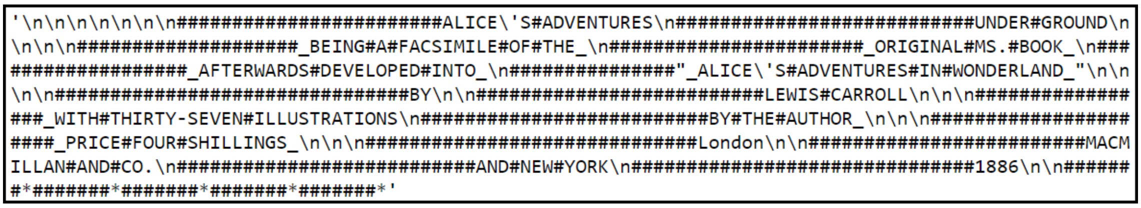


```
          ALICE'S ADVENTURES
             UNDER GROUND



     _BEING A FACSIMILE OF THE_
         _ORIGINAL MS. BOOK_
    _AFTERWARDS DEVELOPED INTO_
"_ALICE'S ADVENTURES IN WONDERLAND_"



                 BY

            LEWIS CARROLL


  _WITH THIRTY-SEVEN ILLUSTRATIONS
            BY THE AUTHOR_


       _PRICE FOUR SHILLINGS_


               London

          MACMILLAN AND CO.
            AND NEW YORK
                1886

*       *       *       *
```

Figure 2.3: Cover page of the book “Alice’s Adventures Under Ground” in data set

Figure 2.3 shows the cover page of the book *“Alice’s Adventures Under Ground”* in the project data set. A few data quality issues can be identified from this example.

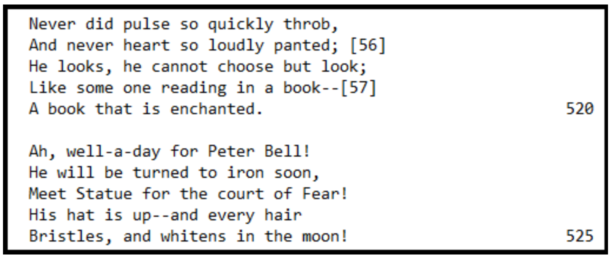
- Alphabets characters are written in capital letters and lower case letter. This reminds us to standardise the characters to lower case to avoid multiple categories of same character.
- The “_” at the beginning and end of the sentence are redundant.
- The “*” or any miscellaneous symbols used as separation lines.
- Consecutive blank spaces and newline characters (“\n”) is perhaps the severest data quality issues. Computer process text differently than human. “\n” is a control character to notify computer for end of the current line and initiation of a new line. Therefore, “\n” and blank spaces are commonly used for text alignment especially in poem and cover page.



```
'\n\n\n\n\n\n\n########################ALICE\'S#ADVENTURES\n###########################UNDER#GROUND\n
\n\n\n####################_BEING#A#FACSIMILE#OF#THE_\n#######################_ORIGINAL#MS.#BOOK_\n###
################_AFTERWARDS#DEVELOPED#INTO_\n###############"_ALICE\'S#ADVENTURES#IN#WONDERLAND_"\n\n
\n\n#################################BY\n\n#########################LEWIS#CARROLL\n\n\n##############
###_WITH#THIRTY-SEVEN#ILLUSTRATIONS\n#########################BY#THE#AUTHOR_\n\n\n##################
####_PRICE#FOUR#SHILLINGS_\n\n\n#############################London\n\n#########################MACM
ILLAN#AND#CO.\n###########################AND#NEW#YORK\n###############################1886\n\n######
#*#######*#######*#######*#######*'
```

Figure 2.4: Raw characters extracted from figure 2.3. Blank spaces are replace with "#" symbol

Figure 2.4 shows the characters extracted from figure 2.3. The spaces are replaced with "#" symbol for greater clarity. Here, the probability of another blank space after current blank space is 90%; Similarly, probability of another '\n" after a "\n" is 56%. Any person with basic English knowledge knows it is implausible.



```
Never did pulse so quickly throb,
And never heart so loudly panted; [56]
He looks, he cannot choose but look;
Like some one reading in a book--[57]
A book that is enchanted.                                520

Ah, well-a-day for Peter Bell!
He will be turned to iron soon,
Meet Statue for the court of Fear!
His hat is up--and every hair
Bristles, and whitens in the moon!                       525
```

Figure 2.5: Reference and Line numbering in the data set

Another data quality issue is the use of line numbering and reference in brackets as shown in figure 2.5. These text are non-original author content and do not contain any stylometry information of the author.

2.3 Data Preparation

Data preparation is the process of feature selection, consolidating, cleaning and transformation of the data. It is next phase after data understanding in CRISP-DM framework. Exploratory data analysis in earlier section provides an insight about the character distribution and the data quality of the data set. This section aimed to address the data quality issues in section 2.2.1 and 2.2.2 and subsequently prepare the data set for modelling.

Python programming language is used as this project due to the fact that extensive support libraries such as scikit-learn, NumPy and Pandas. As the project data set consists of 585 ebooks from 20 authors, one ebook from each author is used for testing while the rest of the ebooks are used for training.

The set of characters determines number of states in Markov chain. It is therefore of paramount importance to reduce the number of characters used in Markov chain from the pool of 165 characters, either by grouping or removing. We start by standardise all the alphabet characters to lower case characters. As there are 60 types of non-English alphabets but only took up less than 0.008% of the total characters, it is therefore too costly to consider each of these to be a single state in Markov chain. Besides that, it is likely these non-English characters incurred additional cost of zero probability event when it appears in testing data but not training data. Because of its low frequency, grouping these non-English characters into a group of “other alphabets” retain most of the information while keeping lowering the negative impact and computing power requirement. Here, the notation “ö” is used for “other alphabets” category for better interpretation.

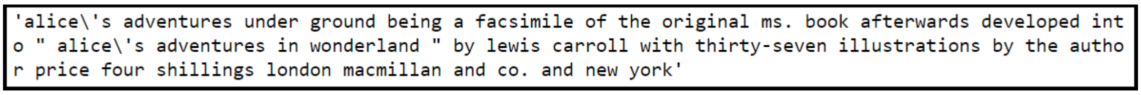
Similarly, there are 65 types of symbols in the project data set. The seven highest frequency symbols shown in table 2.4 were considered in constructing the Markov chain. As mentioned in section 2.2.2, minority symbols are frequently found in various part of the data set as separation lines and page formatting. Likewise, numeric characters are minority characters that made up 0.2% of the text and does not carry much stylometry information. The data quality issues caused by minority numeric and symbol characters can be addressed by replacing them with blank spaces. These characters were replaced instead removed to keep the separation between words.

The functionality of newline character and blank space in the text are rather similar; both of them are used as separation between words. Thus, to address the issue of consecutive newline character, it was replaced with blank space.

Lastly, consecutive blank spaces that was originally present and the additional blank spaces that created in the cleaning process were merged into a single blank space character. In Python, this can be done by with the code *string.replace(“\n”, “ ”)* and *“ ”.join(string.split())*. The earlier code replaced all the newlines with blank spaces. The code *string.split()* without specifying any parameter split the whole string by blank spaces, then “”.join() rejoin all word with a single blank space. The data cleaning procedures are summarised in table 2.5. Figure 2.6 and 2.7 illustrate the example of a post-clean text.

	Input Characters	**Example**	**Output Characters**
Alphabet	26 English alphabets	A-Z, a-z	Lowercase a-z
	Other alphabets	ä, á, ö, œ and etc	Group with the notation “ö”
Symbol	Top most frequent symbols except \n	blank space - ’ ” . ,	As it is
	Other symbols	\n § ¡ ¿ ( ) # and etc	Blank space
Numeric	All Numeric values	0-9	Blank space

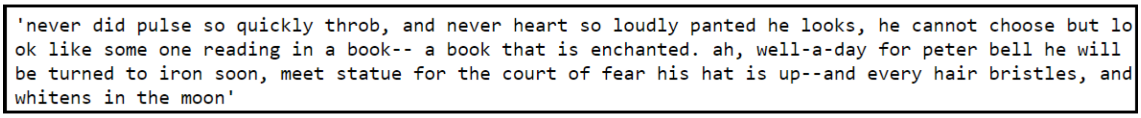
Table 2.5: Data cleaning summary



```
'alice\'s adventures under ground being a facsimile of the original ms. book afterwards developed int
o " alice\'s adventures in wonderland " by lewis carroll with thirty-seven illustrations by the autho
r price four shillings london macmillan and co. and new york'
```

Figure 2.6: Cleaned data from example in figure 2.3

Up to this stage, the project data set (585 books of 20 authors) had been cleaned to the de-



```
'never did pulse so quickly throb, and never heart so loudly panted he looks, he cannot choose but lo
ok like some one reading in a book-- a book that is enchanted. ah, well-a-day for peter bell he will
be turned to iron soon, meet statue for the court of fear his hat is up--and every hair bristles, and
whitens in the moon'
```

Figure 2.7: Cleaned data from example in figure 2.5

sired character format. Next, one book from each author is randomly selected and subsequently split into 10 files with 40,000 characters. This collection of 200 files is the **testing set** used for testing purpose in the project. The remaining of the books (565 books) are used for training. The next step is to compile remaining books of same author in the training into a single file. This reduced the training set to 20 files of length 320,000 to 48,000,000 characters. This training set is named as **training set A**. Lastly, the training set A was split into 10 files each with equal length. The collection of 200 files is named as **training set B**. The data cleaning and preparation procedures are summarised in figure 2.8

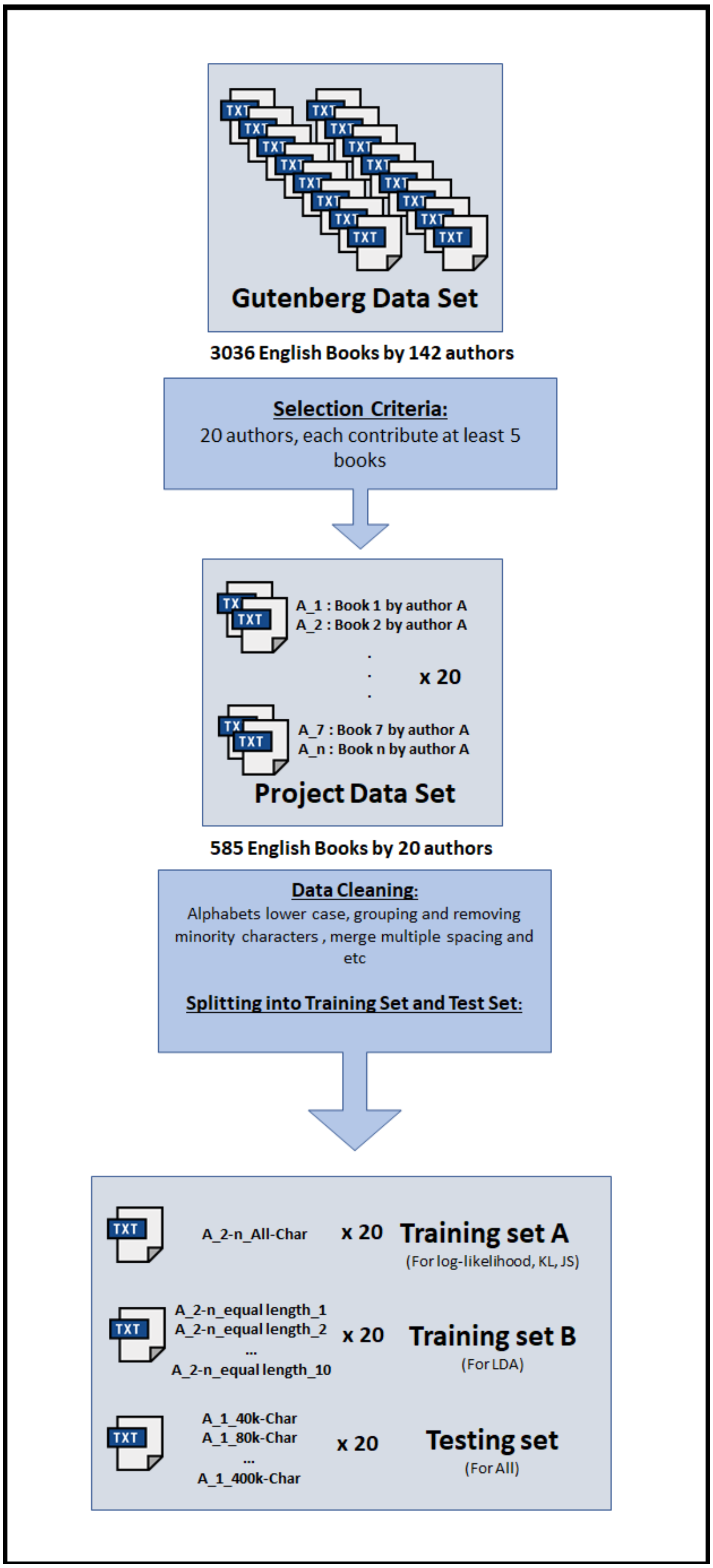


Figure 2.8: Summary of data cleaning and preparation

Chapter 3

Methodology

This chapter resembles the modelling phase in the CRISP-DM framework, defining the modelling techniques used in authorship attribution. The earlier part of this chapter focus on explaining the concept of Markov chain and its mathematical explanation. Subsequently, four similarity measurement methods were proposed with examples. Lastly, the detailed implementation of the model were explained.

3.1 Markov Chain

In most of the time, the outcome of an event is uncertain. For instance, no one can guarantee the outcome of a fair coin flip. Instead, we expressed our prediction by probability - how likely the outcome of an attempt.

The coin flip experiment adhere the concept of independence. In an independent trail process, the knowledge of previous attempt never contribute to the next prediction as every attempt is independence of the other. The series of independence and identical distributed (i.i.i.d) events can be model by Law of Large Number and Central Limit Theorem.

Even so, modeling using the above theorems failed to capture the complexity when there is a correlation between the events. For example, by knowing the position of a football team in league table, we can expect the football team to stay around same the position in the next round. Markov chain (MC) is applicable in this scenario where the future is independent of the past but only the current matter.

Markov chains is a stochastic process that satisfies the Markov property, where the probability of the future state only depends on the current state regardless of the past (Atangana & Gómez-Aguilar 2018). This was introduced by Andres Markov in 1913, where Markov chain were applied to analyse the first 20,000 letter sequence of the poem “Eugeny Onegin” (Von Hilgers & Langville 2006). The transition matrices indicated that the probability of particular letter sequence is a constant and varying between different novel. The probability theory at that time ignored temporal aspect to random event, where the same outcome was obtained from throwing a dice one thousand times and throwing a thousand dice at one times. Markov’s work demonstrated the temporal dependencies that random process depends on the most recent event (Von Hilgers & Langville 2006). Later, Markov chain was first applied in authorship attribution by Morozov (1915) to produced significant discriminant result of different authors.

Because the prediction of future event is independent of the past event, Markov chain is also described as a "memoryless" process. Given a sequence of random variables X with finite states $x_1, x_2, ..., x_n$, the state at time t is called x_t. The conditional probability of the next state given the current state is:

$$P(x_t|x_{t-1}, x_{t-2}, x_{t-3}, \cdots, x_1) = P(x_t|x_{t-1})$$

The different probability can then be assembled to form a transition matrix. The transition matrix has the following properties:

- The transition matrix is a square matrix of size equal to the possible states.
- Since the entries in the transition matrix represent probabilities, the entries are between 0 and 1.
- The row of transition matrix sums up to 1.

Example 1. Supposed a random event has only state A, and B. After three attempts, state A finally changed to state B. We said that Markov Chain has a state space of $S = \{A, B\}$ with a trajectory of $x_0, x_1, x_2, x_3 = A, A, A, B$. Lastly, the Markov chain is said to be at state B at time 3.

Example 2. In this example, Markov chain were used to model customer churn rate of a company. Supposed that a music streaming platform A runs an advertisement campaign. Statistic reports that in every month, this campaign keeps 70% of existing customer to continue the subscription while attracts 60% customers from company B.

Assuming all the customers either subscribe A or B, the possible states are A, B. The state diagram is shown in figure 3.1. The transition matrix P shows the churn rate of the customer:

$$P = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$$

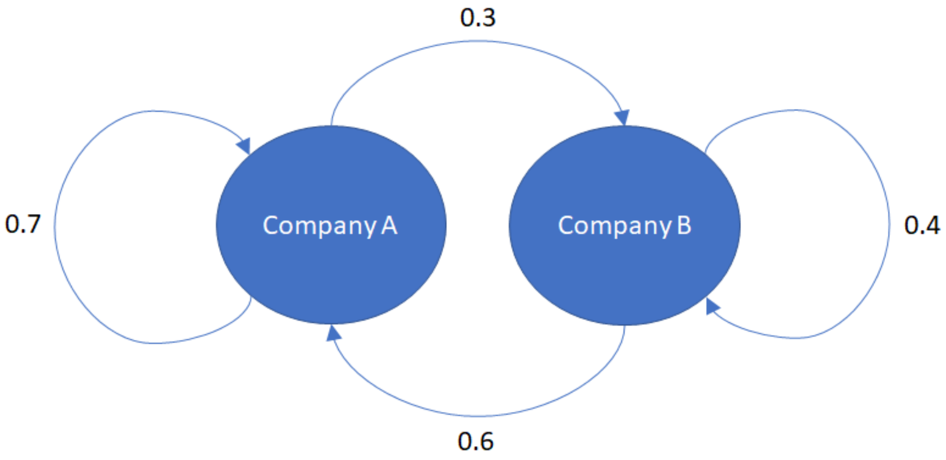


Figure 3.1: State diagram of the churn rate example

Example 3. Continue with the churn rate study in example 2. Before the campaign was launched, company A and company B initially owned 20% and 80% of the market share. Thus, the initial state is $S_0 = [0.2, 0.8]$. The market share of company A and B after one month of campaign, S_1 is:

$$S1 = S_0 \cdot P = \begin{bmatrix} 0.2 & 0.8 \end{bmatrix} \cdot \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} = \begin{bmatrix} 0.46 & 0.54 \end{bmatrix}$$

Similarly, the market share of the companies after two months of launching is:

$$\begin{aligned} S_2 &= S_1 \cdot P \\ &= (S_0 \cdot P) \cdot P \\ &= S_0 \cdot P^2 \\ &= \begin{bmatrix} 0.2 & 0.8 \end{bmatrix} \cdot \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}^2 \\ &= \begin{bmatrix} 0.2 & 0.8 \end{bmatrix} \cdot \begin{bmatrix} 0.61 & 0.39 \\ 0.52 & 0.48 \end{bmatrix} \\ &= [0.538 \quad 0.462] \end{aligned} \tag{3.1}$$

Example 4. Two month the campaign, the management of company A was thrilled seeing a market share hiked from 20% to 53.8% and decided to continue the campaign for another 10 months.

The state after n time steps are the product of initial state and t number of transition matrices. $S_n = S_0 \cdot P^n$. For example, the market share after 6 months is:

$$
\begin{aligned}
S_6 &= S_0 \cdot P^6 \\
&= \begin{bmatrix} 0.2 & 0.8 \end{bmatrix} \quad \cdot \quad \begin{bmatrix} 0.57 & 0.43 \\ 0.57 & 0.43 \end{bmatrix}^6 \\
&= \begin{bmatrix} 0.571 & 0.429 \end{bmatrix}
\end{aligned}
\tag{3.2}
$$

State	**A**	**B**	**n-step probability**
S_0	0.200	0.800	$P^1 = \begin{bmatrix} 0.70 & 0.30 \\ 0.46 & 0.60 \end{bmatrix}$
S_1	0.460	0.540	$P^2 = \begin{bmatrix} 0.61 & 0.39 \\ 0.52 & 0.48 \end{bmatrix}$
S_2	0.538	0.462	$P^3 = \begin{bmatrix} 0.58 & 0.42 \\ 0.56 & 0.44 \end{bmatrix}$
S_3	0.561	0.439	$P^4 = \begin{bmatrix} 0.57 & 0.43 \\ 0.57 & 0.43 \end{bmatrix}$
S_4	0.568	0.432	$P^5 = \begin{bmatrix} 0.57 & 0.43 \\ 0.57 & 0.43 \end{bmatrix}$
S_5	0.571	0.429	$P^6 = \begin{bmatrix} 0.57 & 0.43 \\ 0.57 & 0.43 \end{bmatrix}$
S_6	0.571	0.429	$P^7 = \begin{bmatrix} 0.57 & 0.43 \\ 0.57 & 0.43 \end{bmatrix}$
S_7	0.571	0.429	$P^8 = \begin{bmatrix} 0.57 & 0.43 \\ 0.57 & 0.43 \end{bmatrix}$
S_8	0.571	0.429	$P^9 = \begin{bmatrix} 0.57 & 0.43 \\ 0.57 & 0.43 \end{bmatrix}$
S_9	0.571	0.429	$P^{10} = \begin{bmatrix} 0.57 & 0.43 \\ 0.57 & 0.43 \end{bmatrix}$
S_{10}	0.571	0.429	$P^{11} = \begin{bmatrix} 0.57 & 0.43 \\ 0.57 & 0.43 \end{bmatrix}$
S_{11}	0.571	0.429	$P^{12} = \begin{bmatrix} 0.57 & 0.43 \\ 0.57 & 0.43 \end{bmatrix}$
S_{12}	0.571	0.429	

Table 3.1: Market Share of company A and B after 12 months

Table 3.1 shows the market share of company A and B 12 months after the campaign was launched. Notice that the market share of both companies has stagnant at the fifth month. This is because the Markov chain has converged to its stationary distribution where:

$$S_5 \approx S_6 \approx S_7 \cdots S_n \quad \text{where n larger than 5}$$

A transition matrix is regular if any power of the transition matrix are all positive entries. The transition matrix P used in our churning rate is a *regular matrix*. Markov chain with a regular transition matrix are called *regular Markov chain*. A regular Markov chain in a large number of steps n will reach its stationary or equilibrium state, where for a probability vector π and a transition matrix P, it satisfies:

$$\pi \cdot P^n \approx \pi$$

The stationary state of the regular Markov chain is independent of its initial distribution. That means that regardless what initial percentage of market share of A and B, the regular transition matrix gives the same stationary distribution in a long run. Table 3.2 illustrated same stationary market distribution with different initial distribution using regular transition matrix. It can be seen that the same stationary state were reached in the end. Therefore, regular Markov chain can be applied for long-term prediction.

State	**[A, B]**	**[A, B]**
S_0	[0.000,1.000]	[1.000,0.000]
S_1	[0.400,0.600]	[0.700,0.300]
S_2	[0.520,0.480]	[0.610,0.390]
S_3	[0.556,0.444]	[0.583,0.417]
S_4	[0.567,0.433]	[0.575,0.425]
S_5	[0.570,0.430]	[0.572,0.428]
S_6	[0.571,0.429]	[0.572,0.428]
S_7	[0.571,0.429]	[0.572,0.428]
S_8	[0.571,0.429]	[0.571,0.429]
S_9	[0.571,0.429]	[0.571,0.429]
S_{10}	[0.571,0.429]	[0.571,0.429]
S_{11}	[0.571,0.429]	[0.571,0.429]
S_{12}	[0.571,0.429]	[0.571,0.429]

Table 3.2: Different initial market share achieved same stationary state using a regular transition matrix

3.1.1 Markov Chain order

The discussion so far involves only first order Markov chain, where the next state depends only on the one state preceding it. A higher order Markov chain can be constructed with the same idea. The future state of n-th order Markov chain depends on n-th state preceding it. The n-th order Markov chain is:

$$P(x_t|x_{t-1}, x_{t-2}, x_{t-3}, \cdots, x_1) = P(x_t|x_{t-1}, \cdots, x_{t-n})$$

Thus, the second order Markov chain and third order Markov chain is:

$$P(x_t|x_{t-1}, x_{t-2}, x_{t-3}, \cdots, x_1) = P(x_t|x_{t-1}, x_{t-2})$$

$$P(x_t|x_{t-1}, x_{t-2}, x_{t-3}, \cdots, x_1) = P(x_t|x_{t-1}, x_{t-2}, x_{t-3})$$

Although Markov chain is a “memoryless” process, more memory element can be introduced by using higher order Markov chain to give a better prediction. For instance, if we were asked to predict the next word in the sentence below:

1. “$\cdots$ the ___” *Guess:(odd, wall, apple)*

2. “$\cdots$ against the ___” *Guess:(odd, wall)*

3. “$\cdots$ lean against the ___” *Guess: (wall)*

Thus, a higher order Markov chain can give a better prediction.

3.1.2 Character sequence as stylometry feature

The underlying assumption in any stylometric analysis is every person has a unique writing style which is consistent in their work. The stylometric features served as authorship fingerprint in authorship attribution. In another word, the anonymous text and original writer’s text shares a very similar stylometric features.

The next questions arise is (1) what stylometric features and (2) how do we measure the similarity between features extracted from the text of an anonymous writer and various writers? Our answer for the first question focus on characters-level features. We sought to answer the second question in the next section.

In language context, character is the simplest structure of text. Therefore character-level features is language-independent and can be applied in most of the languages (Neal et al. 2017). On top of that, character-level features are also robust to noises such as spelling inconsistencies and typing errors.

In authorship attribution with character-level features, the author’s work can be viewed as a sequence of characters, where the probability of the character is governed by the writing style. For example, compare to an American author, a British author generally has a higher probability of $P(s|i)$ and a lower $P(z|i)$ due to preference of “ise” ending in words like “organise” and “customise”. In tokenisation process, text were tokenised into token of characters. The transition probabilities can be obtained from the character sequences to form a transition matrix. This transition matrix encodes the writing style of the author.

3.2 Similarity Measurement

This section proposed four attribution models using Log-likelihood (LL), Kullback–Leibler divergence (KLD), Jensen-Shannon Divergence (JD) and Linear Discriminant Analysis (LDA). Lastly, ensemble model with majority voting integrated the prediction of these four models and output its prediction. For better reading experience, this section emphasised on conceptual and mathematical elaboration of the proposed method while the implementation procedure is explained in section 3.3.

3.2.1 Log-likelihood

Definition 3.2.1. Let X be a random variable with probability function $f(x)$. Supposed that a random sample of size k are sampled from this distribution, the *likelihood* of observing $x_1, x_2, \cdots, x_k$ is

$$L(x) = f(x_1)f(x_2)\cdots f(x_k)$$

Example 5. Supposed that a transition matrix of author A, M_A is given and the word "soccer" occurred in the anonymous text.

$$M_A = \begin{array}{c} \\ c \\ e \\ o \\ r \\ s \end{array} \begin{array}{c} \begin{array}{ccccc} c & e & o & r & s \end{array} \\ \begin{pmatrix} 0.1 & 0.2 & 0.3 & 0.15 & 0.25 \\ 0.15 & 0.1 & 0.25 & 0.25 & 0.25 \\ 0.5 & 0.05 & 0.05 & 0 & 0.4 \\ 0.35 & 0.25 & 0.1 & 0.1 & 0.2 \\ 0.4 & 0.2 & 0.05 & 0.05 & 0.3 \end{pmatrix} \end{array}$$

The likelihood of the word "soccer", or the likelihood of sequence of letters "s", "o", "c", "c", "e", "r" is:

$$P(X = soccer) = P(o|s) \cdot P(c|o) \cdot P(c|c) \cdot P(e|c) \cdot P(r|e) \cdot$$

$$P(X = soccer) = 0.05 \cdot 0.5 \cdot 0.1 \cdot 0.2 \cdot 0.25 = 1.25 \times 10^{-5}$$

From the example above, it is clear that the likelihood value can be extremely small even with such small number of characters. This makes the comparison to be difficult in practice. Hence, using *log-likelihood* can be a better alternative.

Definition 3.2.2. Given the likelihood $L(x)$, the log-likelihood $\ell(x)$ is

$$\ell(x) = logL(x) \tag{3.3}$$

Example 6. The log-likelihood for the word "soccer" is:

$$\ell(x) = log(0.05) + log(0.05) + log(0.1) + log(0.2) + log(0.25) = -8.99$$

It is possible to observed a letter sequence in the test set that happened to be a zero-probability event in the transition matrix. For example, the word “czar” is found in one the test set. The letter sequence of “cz” is uncommon in English and did not appear in any of the training set. Therefore, the probability of letter ”z” given “c”, $P(z|c)$ is zero in the training transition matrix.

The likelihood is a joint probability function. As a result, one zero-probability character sequence in the test set can cause the final likelihood to be zero. Similarly, the log-likelihood of zero probability resulting in $-\infty$ log-likelihood of the test set.

It is undesirable but inevitable to encounter a $-\infty$ log-likelihood in the process of identifying the author. The reason why the character sequence appeared in test data but absent in the training data are listed as below:

- The testing data are usually smaller size than training set.
- Typing-error in the test set. Example: “sojzer” instead of “solder”.
- Use of abbreviation in the test set. Example: “qz” instead of “quiz”.
- Use of foreign language.

3.2.2 Kullback–Leibler divergence

Kullback–Leibler divergence (KLD), or simply KL divergence is a difference measurement of one probability distribution to another. KL divergence was originated from information theory to measure information lost when a probability distribution $q(x)$ is used to approximate another probability distribution $p(x)$.

Definition 3.2.3. Given $p(x)$ and $q(x)$, two probability distributions of discrete random variable X. Each distribution sums up to 1, $p(x) > 0$ and $q(x) > 0$ for $x \in X$. The KL divergence, $D_{KL}(p(x)||q(x))$ is:

$$D_{KL}(p(x)||q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)} \tag{3.4}$$

In information theory, KL divergence of $\log 2$ in binary bit system also indicates number of additional bits require to approximate $p(x)$ using $q(x)$. The KL divergence is zero when $p(x) = q(x)$ indicates no information lost. KL divergence is an asymmetric measurement and does not satisfy triangle inequality. Hence, swapping $p(x)$ and $q(x)$ in equation 3.4 does not acquire the same value like in distance measurement.

Example 7. In a deep-sea expenditure, a new species of fish is discovered. The team intended to send the length distribution of the fish to the ground site. Due to bandwidth limitation, the data has to be approximate using simpler distribution with only one or two parameter. Table

3.3 present the available options, binomial distribution $Bin(5, 0.464)$ and uniform distribution $U(0, 5)$.

Using the $p(x)$ as original probability distribution and $q(x)$ as approximate distribution:

Probability distribution			
Length /cm	Original Data $p(x)$	Binomial $q_1(x)$	Uniform $q_2(x)$
0	0.09	0.04	0.17
1	0.14	0.19	0.17
2	0.47	0.33	0.17
3	0.08	0.29	0.17
4	0.10	0.12	0.17
5	0.12	0.02	0.17

Table 3.3: Probability distribution of fish's length data approximation using binomial distribution and uniform distribution

For $q_1(x) = Bin(5, 0.464)$:

$$\begin{aligned} D_{KL}(p(x)||q_1(x)) &= 0.09 \ln \frac{0.09}{0.04} + 0.14 \ln \frac{0.14}{0.19} + 0.47 \ln \frac{0.47}{0.33} + 0.08 \ln \frac{0.08}{0.29} \\ &\quad + 0.10 \ln \frac{0.10}{0.12} + 0.12 \ln \frac{0.12}{0.02} \\ &= 0.29 \end{aligned} \tag{3.5}$$

For $q_2(x) = U(0, 5)$:

$$\begin{aligned} D_{KL}(p(x)||q_2(x)) &= 0.09 \ln \frac{0.09}{0.17} + 0.14 \ln \frac{0.14}{0.17} + 0.47 \ln \frac{0.47}{0.17} + 0.08 \ln \frac{0.08}{0.17} \\ &\quad + 0.10 \ln \frac{0.10}{0.17} + 0.12 \ln \frac{0.12}{0.17} \\ &= 0.24 \end{aligned} \tag{3.6}$$

From the calculation, using uniform distribution $q_2(x) = U(0, 5)$ to approximate the original data $p(x)$ has a smaller KL divergence indicates lesser information lost. Hence, it stands out as a better approximation of the original data.

Similarly, it is inevitable to encounter zero probability letter or word sequence in data mining process. From the equation 3.2.3, zero probability on numerator or denominator of $\ln \frac{p(x)}{q(x)}$ will result in $-\infty$ or undefined value. When $p(x) = 0$, we use the conventions that $0 \cdot \ln \frac{0}{q(x)} = 0$ and $0 \cdot \ln \frac{0}{0} = 0$ (Cover & Thomas 2006). For $q(x) = 0$ and $p(x) > 0$, instead of letting $p(x) \cdot \ln \frac{p(x)}{0} = \infty$ as mentioned by Cover & Thomas (2006), A control experiment is required to decide what value to be assigned. This is because summation of ∞ creates challenge for the classification task. The appropriate value will be discussed in section 4.1.

3.2.3 Jensen-Shannon Divergence

Jensen-Shannon divergence (JD), or JS divergence is a distance measurement between two probability distributions based on the KL divergence.

Definition 3.2.4. Given $p(x)$ and $q(x)$, two probability distributions of discrete random variable X, each distribution sums up to 1. $p(x) > 0$ and $q(x) > 0$ for $x \in X$.

The Jensen-Shannon divergence is:

$$D_{JD}(p(x)||q(x)) = \frac{1}{2} \cdot D_{KL}(p(x)||M) + \frac{1}{2} \cdot D_{KL}(q(x)||M) \tag{3.7}$$

where M is the weighted average of $p(x)$ and $q(x)$. Here, equal weight is assigned to each probability distribution.

$$M = \frac{1}{2} \cdot (p(x) + q(x))$$

JS divergence has the following properties that makes it in favour of KL divergence:

- JS divergence is a positive value.
- The value is bounded. If log_2 base 2 is used, then $0 \leq D_{JD} \leq 1$, where 0 indicates the two distribution is identical and 1 indicates maximum divergence.
- JS divergence is a symmetric distance measurement where $D_{JD}(p(x)||q(x)) = D_{JD}(q(x)||p(x))$. This allow the immediate use of metric index techniques for various application such as similarity search.

Nevertheless, KL divergence is preferable in some situations. KL-divergence asymmetric property can sometimes be an useful advantage in choosing the appropriate $q(x)$ to approximate $p(x)$. For instance, $D_{KL}(q(x)||p(x))$ will be considered when there is a preference to keep the any x that is unlikely to occur in $p(x)$ to also keep unlikely in $q(x)$.

Example 8. Figure 3.4 shows the JS divergence measurement in the example 7:

Probability distribution				
Length /cm	M_1 $0.5(p(x) + q_1(x))$	M_2 $0.5(p(x) + q_2(x))$	$D_{JD}(p(x)\|\|q_1(x))$	$D_{JD}(p(x)\|\|q_2(x))$
0	0.065	0.13	0.005	0.006
1	0.165	0.155	0.002	0.001
2	0.4	0.32	0.006	0.037
3	0.185	0.125	0.032	0.008
4	0.11	0.135	0.000	0.005
5	0.07	0.145	0.020	0.002
		$\sum \mathbf{D_{JD}}$	**0.065**	**0.059**

Table 3.4: Distance measurement of two distribution in table 3.3 using JS divergence

The fact that uniform distribution, $p_2(x)$ yield a lower JS divergence shows that lower information lost when it is used to approximate the original data $p(x)$. In another word, $q_2(x)$ has a lower distance (or higher similarity) to $p(x)$.

3.2.4 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a feature extraction and dimension reduction technique (Ye et al. 2005). It is commonly used to reduce the large dimensional space into smaller dimensional space with good class-separability.

The curse of dimensionality refers to the various problem arise with too many input variables (or dimensions) (Bellman 1966). Model trained with a large set of variables leads to overfitting, where the model failed to learn the underlying pattern. Thus, dimension reduction technique are commonly used to avoid overfitting.

The original linear discriminant was first introduced by Fisher (1936) to solve two-class problem. Rao (1948) later extended the idea to to solve multi-class discriminant analysis in year 1948. Figure 3.2 shows the steel rolling temperature data (Kramer and Jensen 1969) is not linearly separable in x-axis and y-axis figure 3.2, but linearly separable if projected on the right axis in figure 3.3 . LDA assumed the multivariate attribute to have normal distribution with common co-variance Guo et al. (2007).

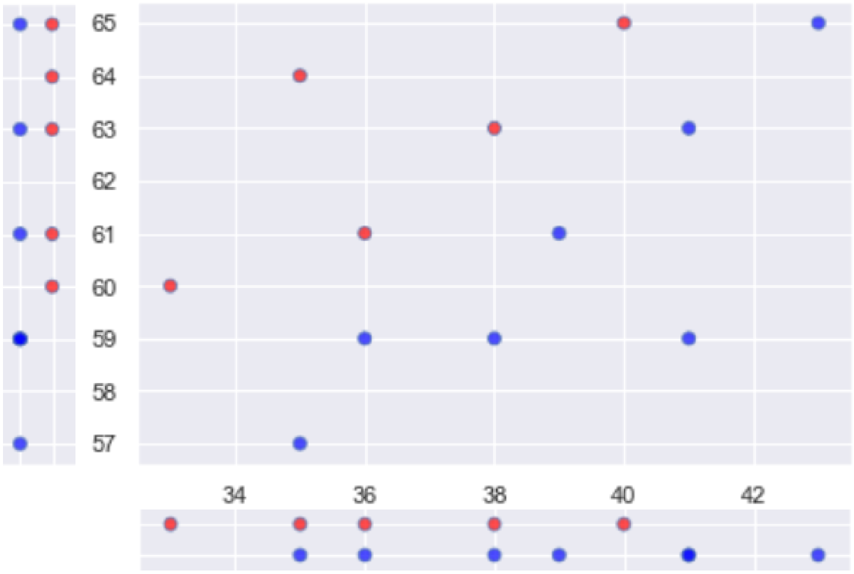


Figure 3.2: Steel rolling temperature data is not linearly separable in x-axis or y-axis

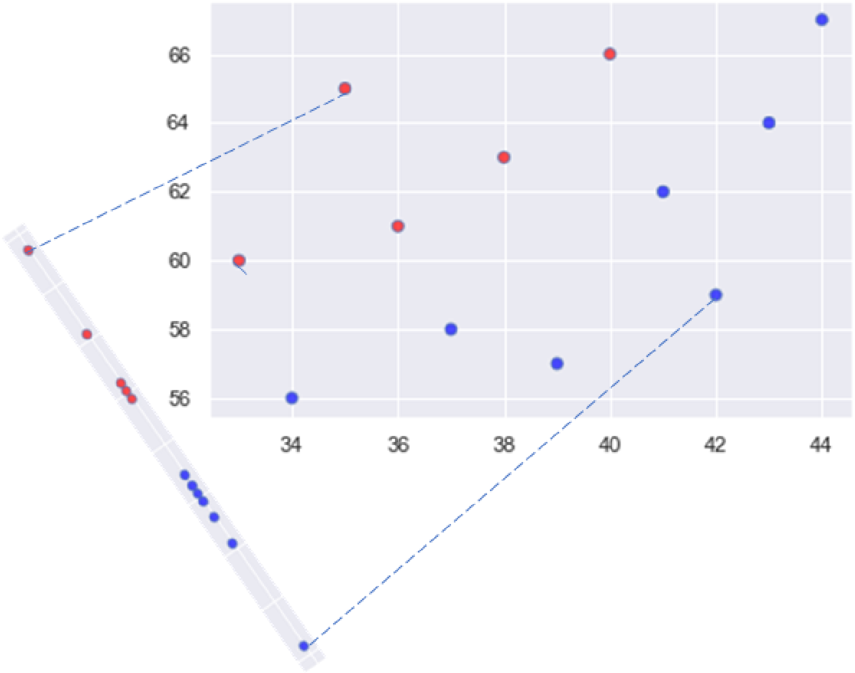


Figure 3.3: Steel rolling temperature data is linearly separable if projecting at the right plane

Two-class LDA

The idea of two-class LDA is to reduce the dimension from bi-variate data x to uni-variate data y through transformation. Suppose that there is a set of m samples with p variables (or dimension), $x_1, x_2, \cdots, x_m$ where $x_i = (x_{i1}, \cdots, x_{ip})$. The samples are two-classes, c_1 and c_2. The scatter matrix of the class S_i is

$$S_i = \sum_{x \subset c_i} (x - \bar{x}_i)(x - \bar{x}_i)' \tag{3.8}$$

where the mean is $\bar{x}_i = \frac{1}{m} \sum_{x \subset c_i}$ x. m_i is the number of samples in the class c_i.

The intra-class scatter matrix or within-class scatter matrix is defined as

$$\hat{\Sigma}_w = S_1 + S_2 = \sum_i \sum_{x \subset c_i} (x - \bar{x}_i)(x - \bar{x}_i)' \tag{3.9}$$

The inter-class scatter matrix or between-class scatter matrix is defined as

$$\hat{\Sigma}_b = (\bar{x}_1 - \bar{x}_i)(\bar{x}_1 - \bar{x}_i)' \tag{3.10}$$

The optimisation problem to find a set of linear combination with a transformation coefficient ϕ that maximise the ratio of of determinant of the $\hat{\Sigma}_b$ and determinant of $\hat{\Sigma}_w$.

$$\Phi = \underset{\Phi}{\text{argmax}} = \frac{det(\Phi^T \hat{\Sigma}_b \Phi)}{det(\Phi^T \hat{\Sigma}_w \Phi)} \tag{3.11}$$

It can be solved by

$$\hat{\Sigma}_b \Phi = \lambda \hat{\Sigma}_w \Phi \tag{3.12}$$

The transform coefficient Φ (eigenvector) is sorted with the highest eigenvalue. The first eigenvector is called linear discriminant 1 (LD1) and carry the most information.

Multi-class LDA

Multi-class LDA is an extension of the standard two-class LDA that is applied in when there are more than two-classes. The

The intra-class scatter matrix or within-class scatter matrix is defined as

$$\hat{\Sigma}_w^n = S_1 + \cdots + S_n = \sum_{i=1}^{n} \sum_{x \subset c_i} (x - \bar{x}_i)(x - \bar{x}_i)' \tag{3.13}$$

The inter-class scatter matrix or between-class scatter matrix is defined as

$$\hat{\Sigma}_b = \sum_{i=1}^{n} m_i (\bar{x}_1 - \bar{x}_i)(\bar{x}_1 - \bar{x}_i)' \tag{3.14}$$

where m_i is the sample size of each class, $\hat{x}_i$ is the mean of each class and $\hat{x}_i$ is the total mean vector given by $\hat{x} = \frac{1}{m} m_i \hat{x}_i$
Similarly, we want to maximised the equation in 3.11. It can be solved using the same equation 3.12.

Classification

After the ϕ transformation is found, classification is done on the transformed space. The transformed input data is classified with the nearest euclidean distance of centroid of the k-th class.

3.2.5 Ensemble Model

Life is full of tough decisions, especially when the decision associate with important financial, medical, privacy and security implication. It is human nature that we collect information and opinions from various sources and experts before reaching a final conclusion. For instance, reading user reviews before making an online purchase, browse several shops before purchasing goods, seek second opinion after doctor's diagnosed of serious illness. Based on our experience and trust to the source, we assign weights to these opinions and integrate these weighted opinion to produce the outcome. This approach is later been applied in classification problem.

Ensemble model is the process where multiple models are used from various modeling algorithm or different training data to output one final decision (Kotu & Deshpande 2015). Although there are multiple models within the ensemble model, it processed the predictions from the collection of models to output a single prediction. The ensemble model is therefore treated as a single performing model. The winning team for KDD cup orange challenge and Netflix grand prize had proved ensemble model to be a high performance model selection (Niculescu-Mizil et al. 2009, Töscher & Jahrer 2009).

Polikar (2006) concluded the reasons people prefers ensemble model. First, ensemble model is preferred because of statistical reasons. In classification problem, a good performing classifier is not guaranteed to perform well in testing data. This is because classifiers are trained on training data but failed to generalised on unseen testing data. Therefore, although sometimes ensemble model do not produce high performance as the member model in its combination, it reduces the risk of picking a bad classifier by chance. This is the reason patient seeks for second opinion to reduce the risk of misdiagnosed or making an decision where other experts think it is not the best. In fact, this is exact reason ensemble model was used in this project.

Other reason ensemble model are used including enormous data volume, too little data, complex problem that requires multiple classifiers to solve, and heterogeneous features of data (Polikar 2006). In any classification task, sufficient and representative data is a pivotal role in making success prediction. When the volume of data are too large to be effectively handle by a single classifiers, it is more efficient to use smaller subset to train multiple classifiers and subsequently create the ensemble. As oppose of enormous data, resampling technique to used to generate additional data and train with additional classifier to form an ensemble. Ensemble model is also proven to be increase the classifier performance in class imbalance situation (Gao et al. 2020, Qian et al. 2014).

Beside data availability factor, ensemble model is also used when the problem is too complex to handle by single classifier. Single linear classifier cannot learn the non-linear decision boundary. Thus, use of a few linear classifiers that generate circular shaped boundaries with majority voting ensemble model can approximate complex decision boundary. In addition, when the features from various data sources are very different in nature, such heterogeneous features requires ensemble of multiple classifiers. In medical diagnosis, ensemble model output the diagnosis result using different classifiers that trained on heterogeneous features such as blood test, urine test and MRI scan.

Majority voting is one of the simple ensemble combination technique. It is sometimes called "plurality voting" in some articles. Majority voting apply equal weights to every member model in ensemble model. The model outputs the prediction that was voted by at least half of the model combination. In the situation that none of the prediction get the simple majority, the ensemble model is said to be failed to produce a stable prediction for this instance. An alternative is to output the prediction that has highest number of votes.

3.3 Modelling and System Design

This section explained the detail implementation of the authorship attribution model. The attribution task can be summarised into three main steps; (1) Feature extraction of the training data and testing data, (2) Similarity measurement or likelihood measurement and subsequently assign the authorship to the author with highest similarity or likelihood (3) Prediction integration with ensemble model.

3.3.1 Feature extraction

The finite number of characters in the data determines the number of step while the set of characters determine the number of state. In first order Markov chain, there are a total of 33 states (26 English alphabets, other-alphabets and 6 symbols). Every character in the training data falls into one of the states.

First, the words in the training data and testing data are tokenised into token of character. Sequence of two characters were sampled and records its number occurrence in the the matrix. The row of the matrix represents the current state and the column represents the next state. Therefore, For p states, first order Markov chain has a matrix of size $p \times p$. The transition matrix can be obtained by dividing the row with the sum of the row in the matrix.

Likewise, second order and third order Markov chain transition matrix can be obtained. In second order Markov chain, the sequence of three characters were sampled, where the first two characters are the current state and the last character is the next state. Thus, we have a transition matrix of size 1089×33. Similarly, third order Markov chain sampled sequence of four characters, using the first three letter as current state and last character as next state. Transition matrices for is obtained from Training set A ($\mathrm{M}_{\text{Training A}}$), Training set B ($\mathrm{M}_{\text{Training B}}$) and Testing set ($\mathrm{M}_{\text{Testing}}$).

3.3.2 Similarity or likelihood method

Log-likelihood Model

In log-likelihood model, character sequences in testing set are view as an observed event. The probability of these character sequences varies between 20 authors and forming 20 different transition matrices.

The testing data were preprocessed in the data preparation phase. It consists of 200 text files each with 40,000 characters length. A single text files in the testing data and transition matrix of one author $\mathrm{M}_{\text{Training A}}$ are used to demonstrate the procedure. First, the given word in the text file are tokenised into characters. The next step is to obtained the log-likelihood of every characters sequences in the testing set using the $\mathrm{M}_{\text{Training A}}$. As a result, 39,999 log-likelihood values were obtained for 40,000 characters. Lastly, the log-likelihood of given text written by authors A is the sum of all the log-likelihood values. Likewise, the log-likelihood of this text written by other authors in the training set can be obtained by using other transition matrices in $\mathrm{M}_{\text{Training A}}$.

It is unavoidable to encounter some character sequences in testing data might not be appeared in author's work especially when transition matrices in used are not the real author. Consequently, the likelihood of this particular sequences written by this author is zero, and the log-likelihood is $log(0) = -\infty$. As a result, log-likelihood of the whole text as a summation function goes to $-\infty$ and consequently "hard" deny the authorship.

In order to avoid $-\infty$ in log-likelihood measurement, the initial mitigation plan is to assign zero for this character sequence. Keep in mind that the likelihood is ranging from 0 to 1 makes log-likelihood ranging from $ln(0) = -\infty$ to $ln(1) = 0$. It is controversial to assigned zero (most likely event) to a $-\infty$ (least likely event). However, by assigning a zero value we ignore this particular letter sequence in the prediction. It exists as a potential problems in our analysis. Particularly, if the given text are short, the weightage of every characters sequences are much higher; Thus, the accumulation of "false 100% probability" has higher negative impact in the prediction. Besides, even with adequate testing data, as the size of transition matrix grow with higher order Markov chain, there are more zero-probability instance in the transition matrix. Thus, increasing the chances of introducing "false probability".

Although the testing data in our project has considerably large size (40,000 characters) to afford having a few of these errors, the workaround for this issue is to use the *log-likelihood per character* that excludes the occurrence of zero log-likelihood in the character sequence. For example, if the 1 zero log-likelihood character are found in characters of size 10, the log-likelihood per character is $\frac{\sum loglikelihood}{10-1}$ This also allow comparison between log-likelihood approach with different testing size. The model then output the prediction with the author that has the highest log-likelihood per character.

KL divergence and JS divergence Model Design

The explanation of JSD and KLD model design was compiled into one section because both model shares the same procedure. The KL divergence and JS divergence of two probability can be obtained using the equation 3.4 and 3.7.

Probability distribution of testing data was chosen to be $p(x)$ the while probability distribution of training data was chosen to be $q(x)$. The fact that $p(x) = 0$ is preferable than $q(x) = 0$

in the equation 3.4 because it is likely to have more zero-probability instance in the transition matrix of testing data. For JS divergence, it is a symmetric distance measurement based on KL divergence. It will used the final $\mathrm{D}_{\mathrm{KL}\ \mathrm{q(x)}}$ value.

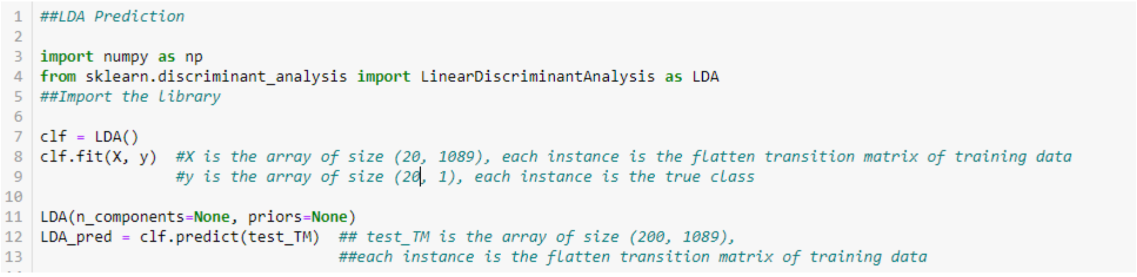
The size of the transition matrices obtained from training and testing data are the same using the same Markov chain order. Therefore the divergence of two transition matrices can directly be compared using the divergence metric. First, the KL divergence of every row in the transition matrices of training set A ($\mathrm{M}_{\mathrm{Training\ A}}$) and testing set ($\mathrm{M}_{\mathrm{Testing}}$) were calculated. Here, $q(x)$ is the row of $\mathrm{M}_{\mathrm{Training\ A}}$ and $p(x)$ is the row of $\mathrm{M}_{\mathrm{Testing}}$. Lastly, the total of KL divergence is obtained by adding all the row KL divergence value. The result can be seen as how diverse the writing style between the unknown author of the test data and the author of the training set. The same procedures were repeated for single $\mathrm{M}_{\mathrm{Testing}}$ in testing set and all the transition matrices $\mathrm{M}_{\mathrm{Training\ A}}$ in training set. Authorship is granted to the author that has the lowest total of KL divergence.

KL divergence is a asymmetric measurement, where $D_{KL}(p(x)||q(x)) \neq D_{KL}(q(x)||p(x))$. Zero-probability in $p(x)$ or $q(x)$ will induce complication in KL divergence calculation. In such case, Cover & Thomas (2006) suggest using the convention of $D_{KL} = 0$ when $p(x) = 0$ and $D_{KL} = \infty$ when $p(x) > 0, q(x) = 0$. However, infinity value of D_{KL} is undesirable because the total divergence can easily go infinity if any of the instance has $p(x) > 0, q(x) = 0$.

Thus, a control experiment using non-overlapping 10 authors, each contribute at least 5 books from Gutenberg data set to find the suitable $\mathbf{D}_{\mathbf{KL\ q(x)}}$ value at different order. The obtained $\mathbf{D}_{\mathbf{KL\ q(x)}}$ value at different MC order will be used in the to in this project with when $p(x) > 0, q(x) = 0$. The control experiment result is presented at section 4.1.

3.3.3 Linear Discriminant Analysis Model Design

LDA is a dimension reduction technique that can be used in classification method. Scikit-learn (SK-learn) was used as the analysis tool to prevent reinvention of the wheel. SK-learn is an open source library built on Python for machine learning and data mining purpose. The main structure of the source code are shown in figure 3.4.



```
##LDA Prediction

import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
##Import the library

clf = LDA()
clf.fit(X, y)  #X is the array of size (20, 1089), each instance is the flatten transition matrix of training data
               #y is the array of size (20, 1), each instance is the true class

LDA(n_components=None, priors=None)
LDA_pred = clf.predict(test_TM)  ## test_TM is the array of size (200, 1089),
                                 ##each instance is the flatten transition matrix of training data
```

Figure 3.4: LDA implementation code

In first order Markov chain, the transition matrix of 33 × 33 can be flatten into a 1-D array of length 1089. Thus, it can be viewed as 20 author classes were decided by this 1089 variables. The prediction function output the class of nearest centroid with the input point. The "*explained_variance_ratio_*" method was used to evaluate the corresponding variance or information in each linear discriminant (LD). The graph of two highest variance linear discriminant LD2 vs LD1 was plotted.

3.3.4 Ensemble Model Design

Ensemble model is a model consists of multiple algorithms or data. Majority voting is an ensemble technique that simply output the most votes prediction. Simple majority simply means "greater than half". For simple majority cases where a class gets more than two votes out of four prediction, ensemble model output the class. The background and reason to used was discussed in section 3.2.5. In this section, an ensemble model is built using combination log-likelihood (LL) model, KL divergence (KLD) model, JS divergence (JSD) model and Linear discriminant analysis (LDA) model. These four members model can output up to four different prediction. For this reason, we need to define the output in the situation of no majority votes or two equal majority votes.

The performance of the member models were evaluated in advance to determine the member model with the best performance. The model is set to output the result of this model when in when there is an equal vote. For instance, the model could get 2 classes getting equally two vote, or four classes each of one vote. In such case, ensemble model is set to output the prediction made by the single best performing model. Table 3.5 summarised the decision made by ensemble model base on different output by its member models. Ensemble model of same Markov chain order and different Markov chain order was constructed.

	Log	**KLD**	**JSD**	**LDA**	**Case**	**Action**	**Output**
Test 1	A	A	A	A	Simple Majority	Output majority class	**A**
Test 2	A	A	A	B	Simple Majority	Output majority class	**A**
Test 3	A	A	B	B	Equal majority	Output prediction made by best member model	**A**
Test 4	A	B	A	B	Equal majority	Output prediction made by best member model	**B**
Test 5	A	B	C	C	One majority	Output majority class	**C**
Test 6	A	B	C	D	No majority	Output prediction made by best member model	**B**

Table 3.5: Output of ensemble model with different member model prediction, assuming KL divergence is the highest performance member model.

3.4 Evaluation Metrics

There are many evaluation metrics available in the task of classification. This section highlight a few metrics that are useful in this project. Table 3.6 illustrated the simple confusion matrix of a binary classifier. Confusion matrix is a tool to describe the performance of the classifier on the test set when the true labels are known.

1. *True positives* (TP) are instances that were predicted positive and actual positive
2. *True negative* (TN) are instances that were predicted negative with actual negative
3. *False positives* (FP) are instances that were predicted positive and actual negative
4. *False negatives* (FN) are instances that were predicted negative and actual positive

TP and TN are the desired instances to be maximised as the total of these two tell how many correct predictions were made. On the contrary, FP and FN indicates the false prediction.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	True Positive (TP)	False Positive(FP)
	Negative	False Negative(FN)	True Negative(TN)

Table 3.6: Confusion matrix

$$Accuracy = \frac{TP}{TP + TN + FN + FP} \tag{3.15}$$

Accuracy, as shown in equation 3.15 is perhaps the most intuitive and widely used metric. It tells the fraction of correct prediction made. Evaluation based on accuracy breaks down when the classes are unbalance (Fawcett & Provost 1996). For instance, if the true distribution of a binary classes is 99:1, making a blind prediction of all first class simply gives 99% accuracy. In addition, accuracy assumes TP, FP, FN and TN equally but in actual there are different error cost. For instance, a patient was misdiagnosed in his cancer diagnosis test. A false positive can lead to time and monetary lost, associated with mental stress on the patient. However, a false negative result is life threatening if early treatment was missed.

$$Precision = \frac{TP}{TP + FP}$$

Precision, also known as True Positive Rates (TPR) measures the ratio of the true positive with total positive prediction. It is useful when the false positives are costly. For instance, false positive from authorship attribution model in crime investigation might result the innocence to be wrongly convicted.

$$Recall = \frac{TP}{TP + FN}$$

Recall, or sensitivity is another useful metric that measures the ratio of the true positive with total true positive class. This metric is often use when the cost of false negatives are high. For example, information retrieval of relevant documents in the data base.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

F1 score is the harmonic mean of precision and recall. It measures the classifier's performance by taking account of the false negative and false positives. In addition, F1 score can be useful in class imbalance cases.

Both accuracy and F1-score will be used as the evaluation metric to determine the performance of our classifier in this project. Mainly because equal true label distribution and the project treats equal cost and benefits among true and false predictions. In addition, our project objectives focus on generalised problem where the goal is not to identified specific authors.

Chapter 4

Result and Discussion

This chapter resembles the evaluation phase in the CRISP-DM framework. This chapter starts by presenting the $\mathbf{D_{KL\ q(x)}}$ value in the control experiment. The next part in this chapter elaborated the result of first order and higher order Markov chain model with confusion matrices with individual author class performance. The discussion focus on comparing the model of same MC order and investigate the performance of the model at higher order. Later, evaluation of ensemble model of first order MC and all MC order were discussed. Lastly, few techniques were proposed to mitigate the initial zero-probability distribution.

4.1 KL Divergence Parameter Control Experiment

This section presents the performance of the control KLD model at 1^{st}, 2^{nd}, 3^{rd} MC order with respect to different $\mathbf{D_{KL\ q(x)}}$ value. As mention in section 3.3.2, a control experiment that uses non-overlapping data set was used to investigate the suitable $\mathbf{D_{KL\ q(x)}}$ at different MC order to be assigned when $p(x) > 0, q(x) = 0$.

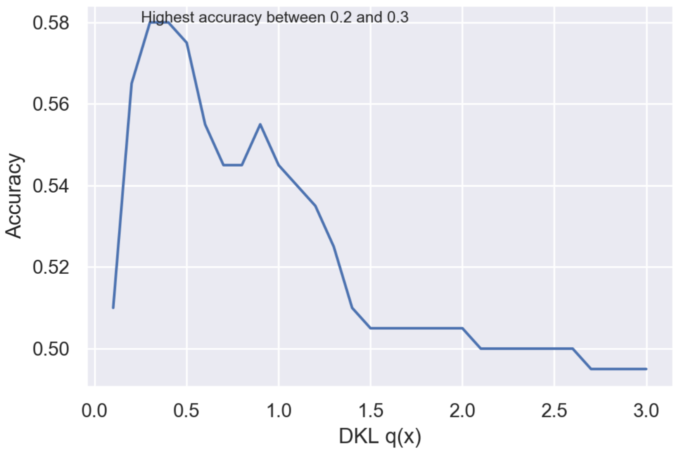


Figure 4.1: 1^{st} *order MC KLD model accuracy with respect to different* $\mathbf{D_{KL\ q(x)}}$

Figure 4.1 shows the 1^{st} order MC KLD model accuracy has the highest accuracy of 58% using $\mathbf{D_{KL\ q(x)}}$ of 0.2 and 0.3. Thus, $\mathbf{D_{KL\ q(x)}}$ = 0.2 will be used in all 1^{st} order MC model. This includes 1^{st} order MC KLD model, 1^{st} order MC JSD model which was built on top of KLD measurement, and 1^{st} order MC ensemble model.

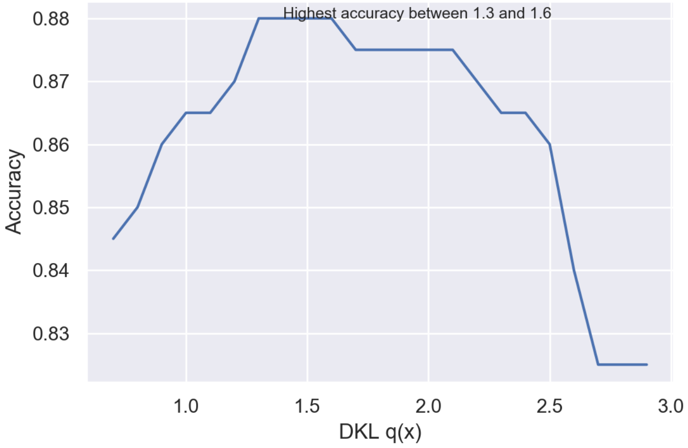


Figure 4.2: 2^{nd} order MC KLD model accuracy with respect to different $\mathbf{D_{KL\ q(x)}}$

Figure 4.2 shows the 2^{nd} order MC KLD model accuracy has the highest accuracy of 88% using $\mathbf{D_{KL\ q(x)}}$ of 1.3 and 1.6. Thus, $\mathbf{D_{KL\ q(x)}}$ = 1.5 will be used in all 2^{nd} order MC model.

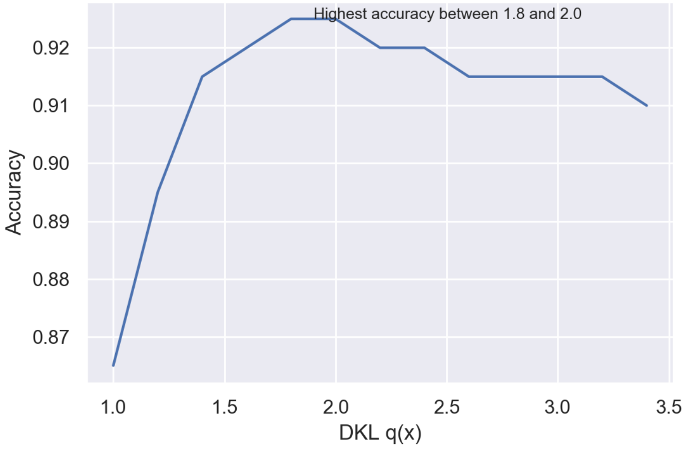


Figure 4.3: 3^{rd} order MC KLD model accuracy with respect to different $\mathbf{D_{KL\ q(x)}}$

Figure 4.3 shows the 3^{rd} order MC KLD model accuracy has the highest accuracy of 92.5% using $\mathbf{D_{KL\ q(x)}}$ of 1.8 and 2.0. Thus, $\mathbf{D_{KL\ q(x)}}$ = 1.9 will be used in all 3^{rd} order MC model.

4.2 First order Markov Chain

4.2.1 First order Markov Chain Log-likelihood Model

Figure 4.4 presents the 1^{st} order MC LL model prediction in confusion matrix. The model had achieved 67% accuracy while baseline result with random guess is 5%. There are a few interesting findings from the confusion matrix in figure 4.4:

1. 1^{st} order MC LL model had a poor performance in identifying certain author. For instance, *Charles Dickens* and *D H Lawrence* do no get any of the prediction.

2. The wrong prediction cases were misclassified into several other classes.

There is one observation that caught our interest. Many test set were systematically misclassified into *Hamlin Garland* and *James F C*. For instance, *DH Lawrance* and *Jack London* were very likely being misclassified as *Hamlin Garland*. On the other hand, *Hamlin Garland* and other three authors were tend to misclassify to *James F C*. Table 4.1 shows *Hamlin Garland* and *James F C* has low precision.

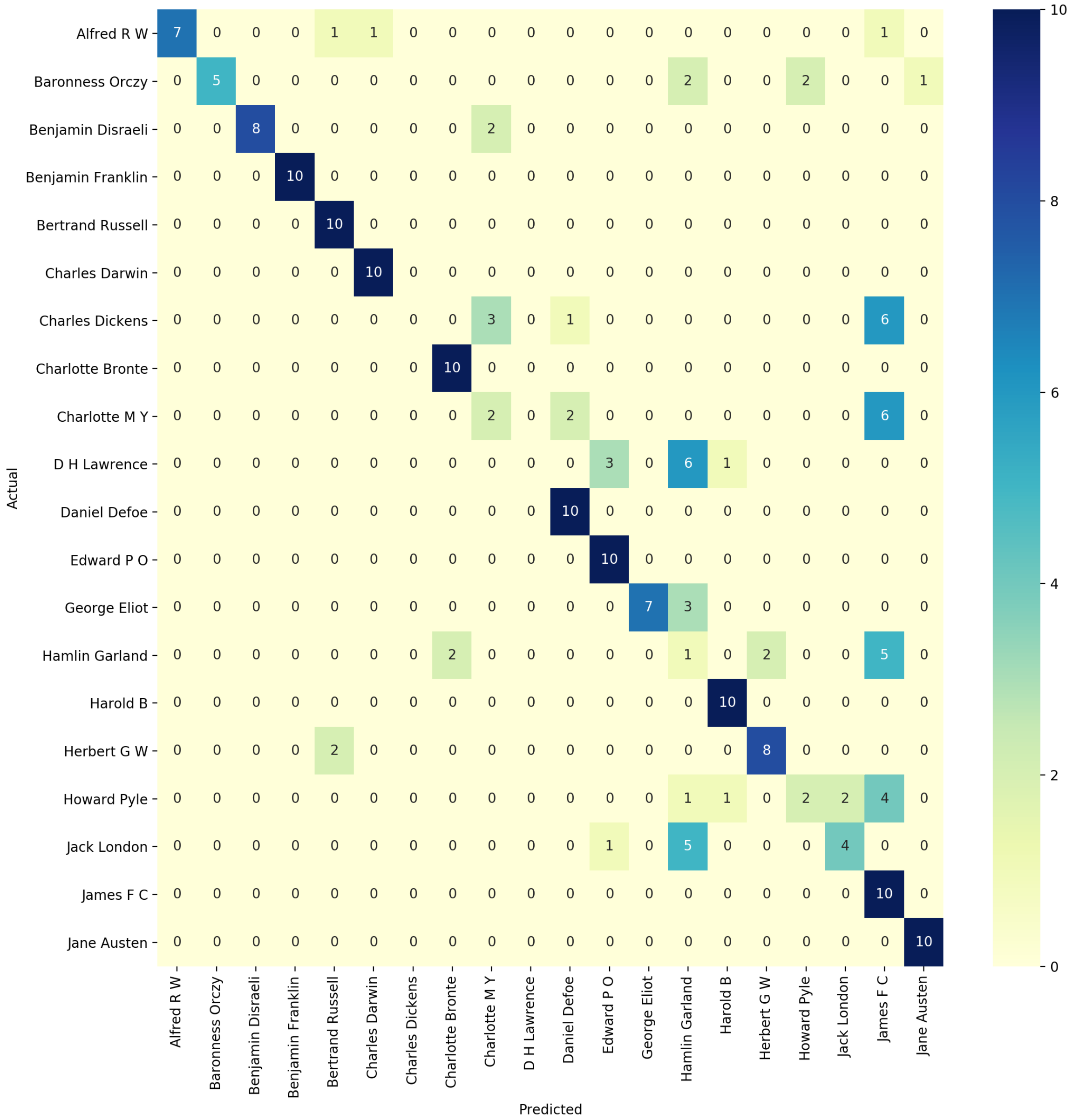


Figure 4.4: Confusion matrix of 1^{st} order Markov chain prediction model with log-likelihood approach

As mentioned in data preparation phase, one book of each author was split into 10 test data each at 40,000 characters. Figure 4.5 shows the log-likelihood per character value using the test set of *Alfred R W*. The value are sorted in descending order and the model output the class with highest log-likelihood per character as the prediction. Although there is $\frac{3}{10}$ were misclassified

Author	**First order Markov Chain**		
	Precision	**Recall**	**F1 Score**
Alfred Russel Wallace	1	0.7	0.82
Baronness Orczy	1	0.5	0.67
Benjamin Disraeli	1	0.8	0.89
Benjamin Franklin	1	1	1
Bertrand Russell	0.77	1	0.87
Charles Darwin	0.91	1	0.95
Charles Dickens	0	0	0
Charlotte Bronte	0.83	1	0.91
Charlotte Mary Yonge	0.29	0.2	0.24
D H Lawrence	0	0	0
Daniel Defoe	0.77	1	0.87
Edward Phillips Oppenheim	0.71	1	0.83
George Eliot	1	0.7	0.82
Hamlin Garland	0.06	0.1	0.07
Harold Bindloss	0.83	1	0.91
Herbert George Wells	0.8	0.8	0.8
Howard Pyle	0.5	0.2	0.29
Jack London	0.67	0.4	0.5
James Fenimore Cooper	0.31	1	0.48
Jane Austen	0.91	1	0.95
Accuracy			0.67
F1 Score	0.67	0.67	0.64

Table 4.1: First order Markov Chain Log-likelihood model performance table

in the confusion matrix, figure 4.5 tells that the model will output correct prediction if the whole book as a single test set. In another word, the model is converging towards its original author, suggesting the misclassification were caused by randomness present in individual test data. The character-level features in 1^{st} order MC using LL method can effective to distinguish this author class.

Figure 4.6 illustrates some of the author classes which its log-likelihood per character did not converge towards its author. In the figure, the true author was highlighted for better clarity. The features extracted from 1^{st} order MC model using 33 states clearly were not enough to distinguish the true author from the candidates. We also observed tiny gap between log-likelihood per characters of each author, indicates the model was not able to significantly discriminate between authors.

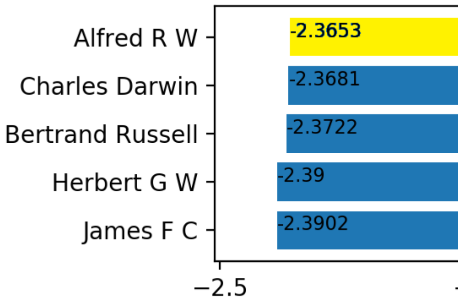


Figure 4.5: Log-likelihood per character (descending order). Truth author is highlighted in yellow

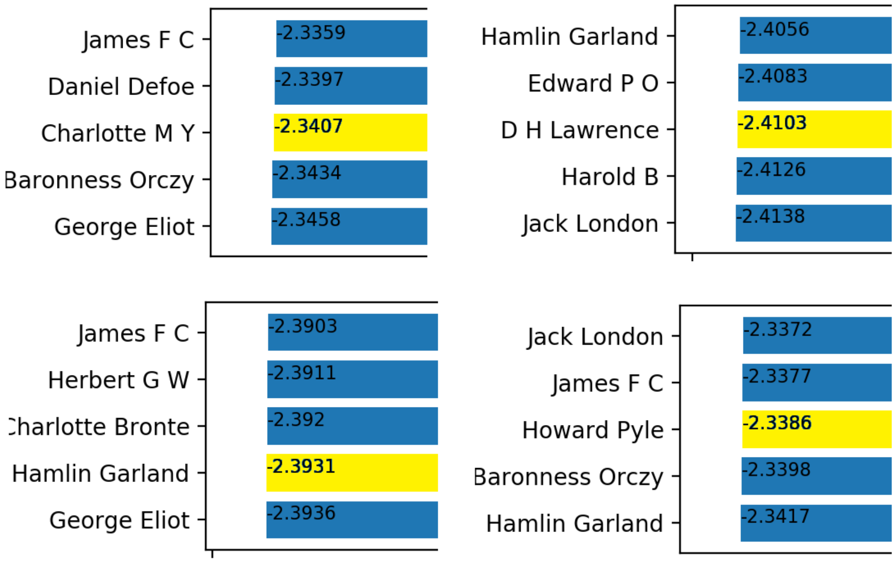
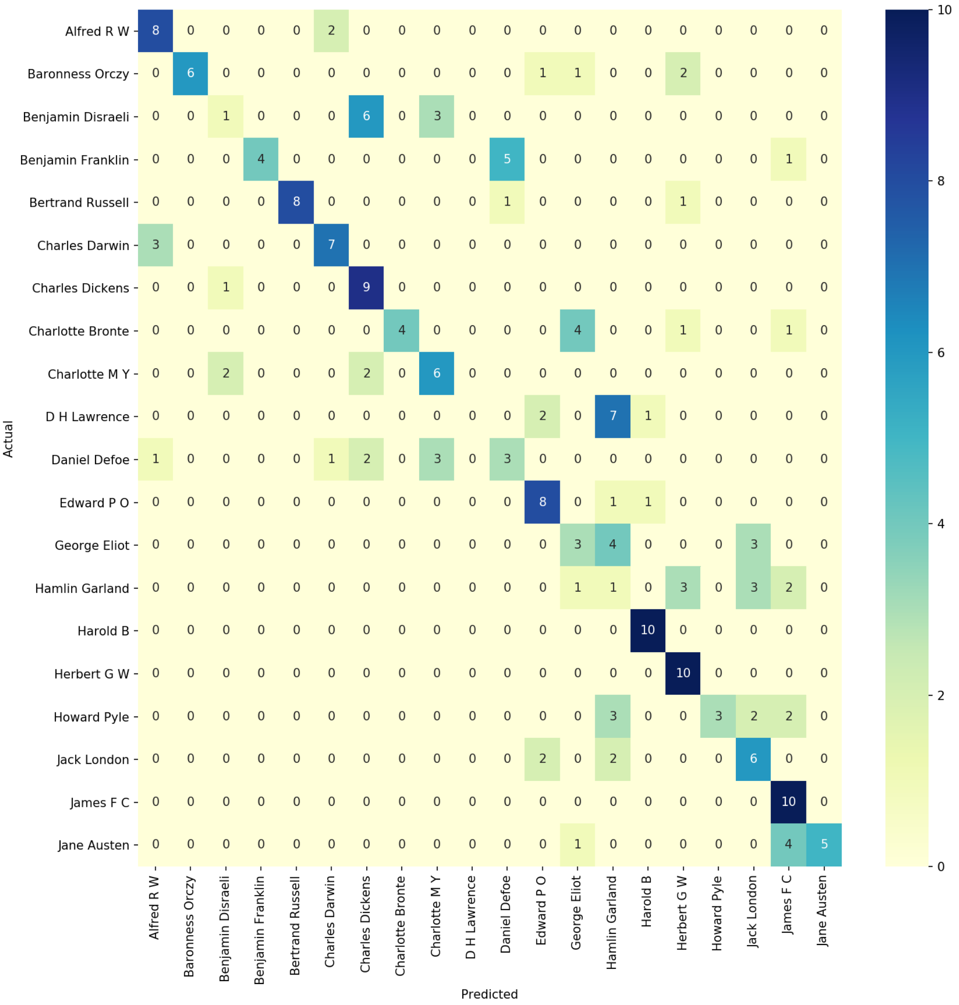


Figure 4.6: Log-likelihood per character (descending order). Truth author is highlighted in yellow

4.2.2 First order Markov Chain KLD and JSD Model

First order Markov Chain KL Divergence Result Evaluation

Figure 4.7 shows the confusion matrix of the 1^{st} order MC KLD model using $\mathbf{D}_{\mathbf{KL\ q(x)}} = 0.2$. The model has achieved 58% accuracy while baseline result with random guess is 5%.



	Alfred R W	Baronness Orczy	Benjamin Disraeli	Benjamin Franklin	Bertrand Russell	Charles Darwin	Charles Dickens	Charlotte Bronte	Charlotte M Y	D H Lawrence	Daniel Defoe	Edward P O	George Eliot	Hamlin Garland	Harold B	Herbert G W	Howard Pyle	Jack London	James F C	Jane Austen
Alfred R W	8	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Baronness Orczy	0	6	0	0	0	0	0	0	0	0	0	1	1	0	0	2	0	0	0	0
Benjamin Disraeli	0	0	1	0	0	0	6	0	3	0	0	0	0	0	0	0	0	0	0	0
Benjamin Franklin	0	0	0	4	0	0	0	0	0	0	5	0	0	0	0	0	0	0	1	0
Bertrand Russell	0	0	0	0	8	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
Charles Darwin	3	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charles Dickens	0	0	1	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0
Charlotte Bronte	0	0	0	0	0	0	0	4	0	0	0	0	4	0	0	1	0	0	1	0
Charlotte M Y	0	0	2	0	0	0	2	0	6	0	0	0	0	0	0	0	0	0	0	0
D H Lawrence	0	0	0	0	0	0	0	0	0	0	0	2	0	7	1	0	0	0	0	0
Daniel Defoe	1	0	0	0	0	1	2	0	3	0	3	0	0	0	0	0	0	0	0	0
Edward P O	0	0	0	0	0	0	0	0	0	0	0	8	0	1	1	0	0	0	0	0
George Eliot	0	0	0	0	0	0	0	0	0	0	0	0	3	4	0	0	0	3	0	0
Hamlin Garland	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	3	0	3	2	0
Harold B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
Herbert G W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0
Howard Pyle	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	2	2	0
Jack London	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	0	0	6	0	0
James F C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0
Jane Austen	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	4	5

Figure 4.7: Confusion matrix of 1^{st} order Markov chain prediction model with KL divergence approach

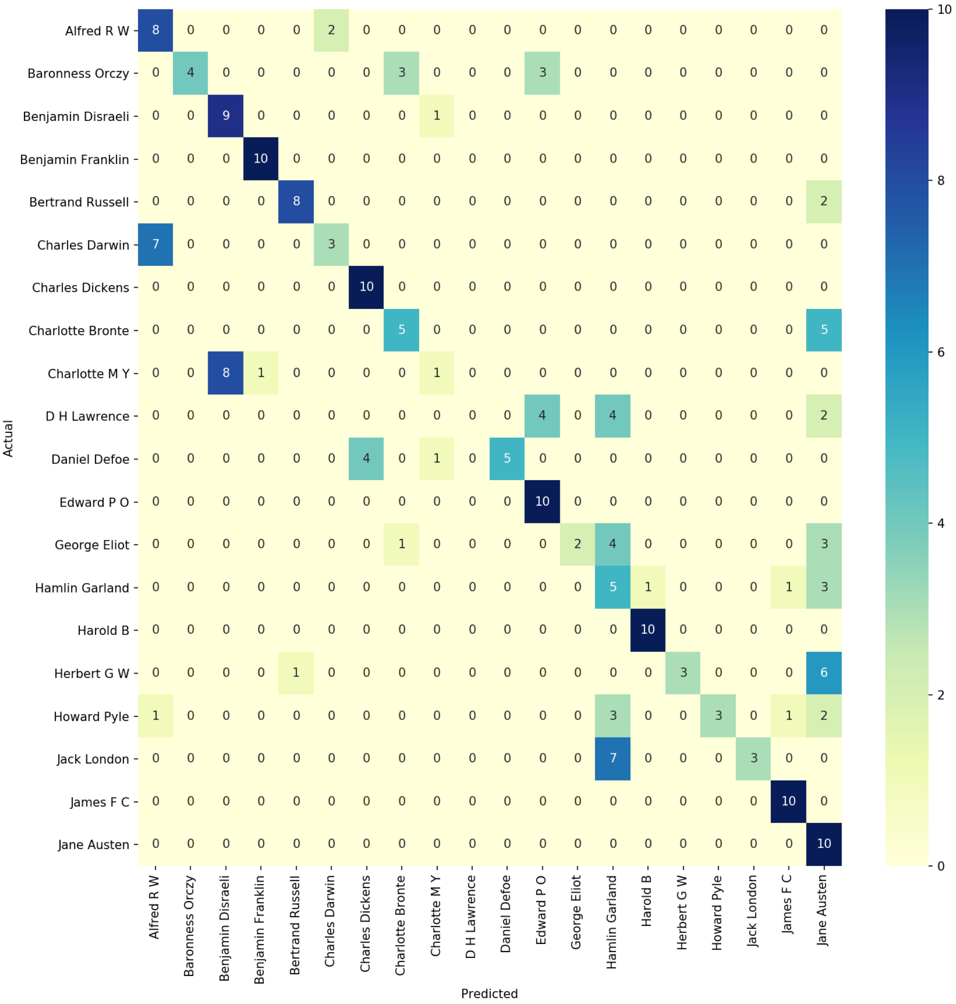
Table 4.2 shows the performance of 1^{st} order MC KLD model by class. KLD model has a poor F1 score of 0.58. Many wrongly predicted classes were sparsely misclassified to several wrong classes resulting the model getting a lower precision among others of same MC order.

The foundation of JSD model was built on top of KLD. In addition, both models measure using the transition matrices of training data and testing data. Such similar nature let us wonder if these two share the same result. In fact, KLD model and JSD model has the same accuracy (0.58) and approximately same F1-score (KLD: 0.56 and JSD: 0.57). The comparison of 1^{st} order MC KLD and JSD model will be discussed in the section 4.2.2.

Author	**Precision**	**Recall**	**F1 Score**
Alfred Russel Wallace	0.57	0.8	0.67
Baronness Orczy	0.9	0.9	0.9
Benjamin Disraeli	0.25	0.1	0.14
Benjamin Franklin	1	0.7	0.82
Bertrand Russell	1	0.7	0.82
Charles Darwin	0.75	0.6	0.67
Charles Dickens	0.17	0.1	0.12
Charlotte Bronte	0.8	0.4	0.53
Charlotte Mary Yonge	0.38	0.5	0.43
D H Lawrence	0	0	0
Daniel Defoe	0.67	0.8	0.73
Edward Phillips Oppenheim	0.8	0.8	0.8
George Eliot	0.38	0.3	0.33
Hamlin Garland	0.08	0.1	0.09
Harold Bindloss	0.82	0.9	0.86
Herbert George Wells	0.69	0.9	0.78
Howard Pyle	0.6	0.6	0.6
Jack London	0.37	1	0.54
James Fenimore Cooper	0.64	0.7	0.67
Jane Austen	0.7	0.7	0.7
Accuracy			0.58
F1 Score	0.58	0.58	0.56

Table 4.2: First order Markov chain KLD model performance table

First order Markov chain JS Divergence Result Evaluation



	Alfred R W	Baronness Orczy	Benjamin Disraeli	Benjamin Franklin	Bertrand Russell	Charles Darwin	Charles Dickens	Charlotte Bronte	Charlotte M Y	D H Lawrence	Daniel Defoe	Edward P O	George Eliot	Hamlin Garland	Harold B	Herbert G W	Howard Pyle	Jack London	James F C	Jane Austen
Alfred R W	8	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Baronness Orczy	0	4	0	0	0	0	0	3	0	0	0	3	0	0	0	0	0	0	0	0
Benjamin Disraeli	0	0	9	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Benjamin Franklin	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bertrand Russell	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Charles Darwin	7	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charles Dickens	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0
Charlotte Bronte	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	5
Charlotte M Y	0	0	8	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
D H Lawrence	0	0	0	0	0	0	0	0	0	0	0	4	0	4	0	0	0	0	0	2
Daniel Defoe	0	0	0	0	0	0	4	0	1	0	5	0	0	0	0	0	0	0	0	0
Edward P O	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
George Eliot	0	0	0	0	0	0	0	1	0	0	0	0	2	4	0	0	0	0	0	3
Hamlin Garland	0	0	0	0	0	0	0	0	0	0	0	0	0	5	1	0	0	0	1	3
Harold B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
Herbert G W	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	3	0	0	0	6
Howard Pyle	1	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	0	1	2
Jack London	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	3	0	0
James F C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0
Jane Austen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10

Figure 4.8: Confusion matrix of 1^{st} order Markov chain prediction model with JS divergence approach

Figure 4.8 shows the confusion matrix of 1^{st} order MC JSD model. The model has achieved 58% accuracy while baseline result with random guess is 5%.

The performance detailed by class is shown in table 4.3. JSD model has a poor F1 score of 0.57. From the confusion matrix, many wrongly predicted classes were sparsely misclassified to several wrong prediction.

Author	**Precision**	**Recall**	**F1 Score**
Alfred Russel Wallace	0.56	0.9	0.69
Baronness Orczy	1	0.7	0.82
Benjamin Disraeli	0.88	0.7	0.78
Benjamin Franklin	1	0.9	0.95
Bertrand Russell	1	0.8	0.89
Charles Darwin	0.8	0.4	0.53
Charles Dickens	0.5	0.2	0.29
Charlotte Bronte	0.8	0.4	0.53
Charlotte Mary Yonge	0.33	0.1	0.15
D H Lawrence	0	0	0
Daniel Defoe	0.88	0.7	0.78
Edward Phillips Oppenheim	0.71	1	0.83
George Eliot	1	0.2	0.33
Hamlin Garland	0	0	0
Harold Bindloss	0.59	1	0.74
Herbert George Wells	0.62	0.5	0.56
Howard Pyle	0.57	0.4	0.47
Jack London	0.8	0.8	0.8
James Fenimore Cooper	0.62	1	0.77
Jane Austen	0.24	1	0.38
Accuracy			0.58
F1 Score	0.58	0.58	0.56

Table 4.3: JS divergence performance table

Performance comparison between 1^{st} order MC KLD and JSD model

JSD model obtained a slightly higher F1 score (0.57) than KL divergence model (0.56). These two model both using KLD measurement and measure based on the transition matrices. In general, KLD model and JSD model has similar performances in most of the classes based on F1 score. From table 4.2 and table 4.3, the main differences between the two models occurred at these four author classes; *Benjamin Disraeli*, *Charlotte Mary Yonge*, *Jack London* and *Jane Austen*.

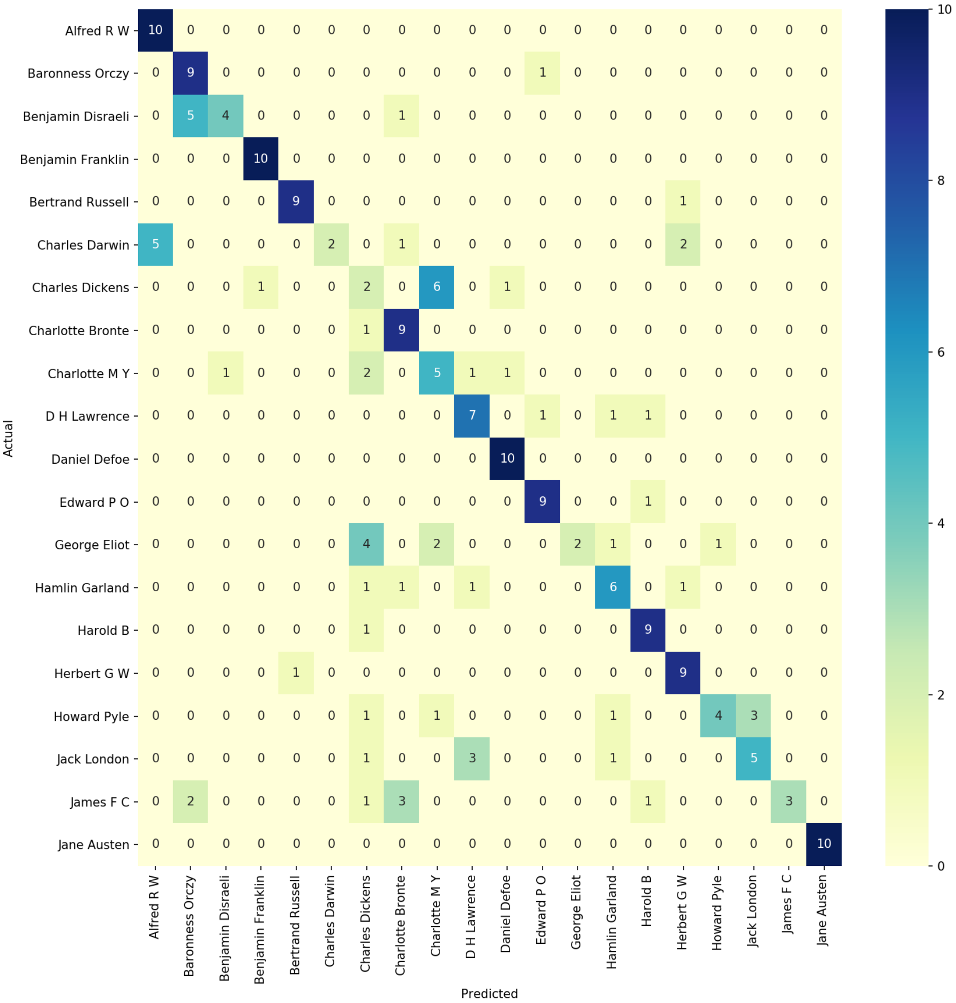
JSD model has a poorer precision in the author class *Charlotte Mary Yonge* and *Jane Austen*. At 1^{st} order MC, JSD model obtained a precision of 0.15 and 0.38 for these two classes while KLD model had a precision of 0.43 and 0.7. This indicates that JSD model makes a lot of false positive prediction in these two classes.

Nevertheless, 1^{st} order MC JSD model is better in classifying the author classes of *Benjamin Disraeli* and *Jack London* as compared to KLD model. JSD has F1 score of 0.78 and 0.8 while the later model only has 0.14 and 0.54. In fact, JSD model has the highest F1 score in the Jack London class among all the member models of the same MC order.

In conclusion, JSD model and KLD model each has its own strength in different author class. Their strength complement each other and able to enhance the prediction performance by ensemble model.

4.2.3 First order Markov Chain LDA Model

Figure 4.9 shows the confusion matrix of the first order Markov chain prediction by LDA model. LDA model is one of the two highest model in 1^{st} order MC with 67% accuracy while baseline result with random guess is 5%. It also obtained a high F1 score of 0.65. The detail performance by class of first order LDA model is shown in table 4.4.



	Alfred R W	Baroness Orczy	Benjamin Disraeli	Benjamin Franklin	Bertrand Russell	Charles Darwin	Charles Dickens	Charlotte Bronte	Charlotte M Y	D H Lawrence	Daniel Defoe	Edward P O	George Eliot	Hamlin Garland	Harold B	Herbert G W	Howard Pyle	Jack London	James F C	Jane Austen
Alfred R W	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Baroness Orczy	0	9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Benjamin Disraeli	0	5	4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Benjamin Franklin	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bertrand Russell	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Charles Darwin	5	0	0	0	0	2	0	1	0	0	0	0	0	0	0	2	0	0	0	0
Charles Dickens	0	0	0	1	0	0	2	0	6	0	1	0	0	0	0	0	0	0	0	0
Charlotte Bronte	0	0	0	0	0	0	1	9	0	0	0	0	0	0	0	0	0	0	0	0
Charlotte M Y	0	0	1	0	0	0	2	0	5	1	1	0	0	0	0	0	0	0	0	0
D H Lawrence	0	0	0	0	0	0	0	0	0	7	0	1	0	1	1	0	0	0	0	0
Daniel Defoe	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
Edward P O	0	0	0	0	0	0	0	0	0	0	0	9	0	0	1	0	0	0	0	0
George Eliot	0	0	0	0	0	0	4	0	2	0	0	0	2	1	0	0	1	0	0	0
Hamlin Garland	0	0	0	0	0	0	1	1	0	1	0	0	0	6	0	1	0	0	0	0
Harold B	0	0	0	0	0	0	1	0	0	0	0	0	0	0	9	0	0	0	0	0
Herbert G W	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0
Howard Pyle	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	4	3	0	0
Jack London	0	0	0	0	0	0	1	0	0	3	0	0	0	1	0	0	0	5	0	0
James F C	0	2	0	0	0	0	1	3	0	0	0	0	0	0	1	0	0	0	3	0
Jane Austen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10

Figure 4.9: Confusion matrix of 1^{st} order Markov chain prediction model with LDA approach

The transform coefficient ϕ, also known as linear discriminant (LD) in equation 3.12 were sort by its eigenvalue. LD1 and LD2 contained the most information and correspond to 20% and 17% variation of the data. To investigate the misclassification on LDA model, we try to relate the class performance with its transformed data point of LD2 vs LD1 in figure 4.10.

	Precision	Recall	F1 Score
Alfred Russel Wallace	0.67	1	0.8
Baronness Orczy	0.56	0.9	0.69
Benjamin Disraeli	0.8	0.4	0.53
Benjamin Franklin	0.91	1	0.95
Bertrand Russell	0.9	0.9	0.9
Charles Darwin	1	0.2	0.33
Charles Dickens	0.14	0.2	0.17
Charlotte Bronte	0.6	0.9	0.72
Charlotte Mary Yonge	0.36	0.5	0.42
D H Lawrence	0.58	0.7	0.64
Daniel Defoe	0.83	1	0.91
Edward Phillips Oppenheim	0.82	0.9	0.86
George Eliot	1	0.2	0.33
Hamlin Garland	0.6	0.6	0.6
Harold Bindloss	0.75	0.9	0.82
Herbert George Wells	0.69	0.9	0.78
Howard Pyle	0.8	0.4	0.53
Jack London	0.62	0.5	0.56
James Fenimore Cooper	1	0.3	0.46
Jane Austen	1	1	1
Overall Performance			
Accuracy			0.67
F1 score	0.73	0.67	0.65

Table 4.4: First order Markov chain LDA model performance table

LDA maximising the ratio of inter-class scatter and intra-class scatter. Hence, in figure 4.10, we observed the centroid of the data points were kept far apart but data points of same class was located closely together. In **cluster A**, the author *Benjamin Russell* (Marker: Yellow X) has great class separation with other authors. In another words, his writing style are uniquely distinct with other 19 authors. Besides, the intra-class scatter is small. This signifies low variation between the writing style and indicates the consistency writing style in his book. Thus, it can be easily distinguish in the transformed space and it has high precision (0.9) and high recall (0.9).

In **cluster B**, *Alfred R W* (Marker: Red X) has low intra-class scatter. Thus, unseen data also inherit his consistent writing style and located near the centroid of its class. Thus, result in table 4.4 shows it has high recall (1). Unfortunately, its precision was lower by its neighbour author that has high in class scatter. From the transformed space, we can see *Charles Darwin* (Marker: Green diamond) has a sparse in-class spread. Hence, in figure 4.9, some of *Charles Darwin* works were misclassified into *Alfred R W*.

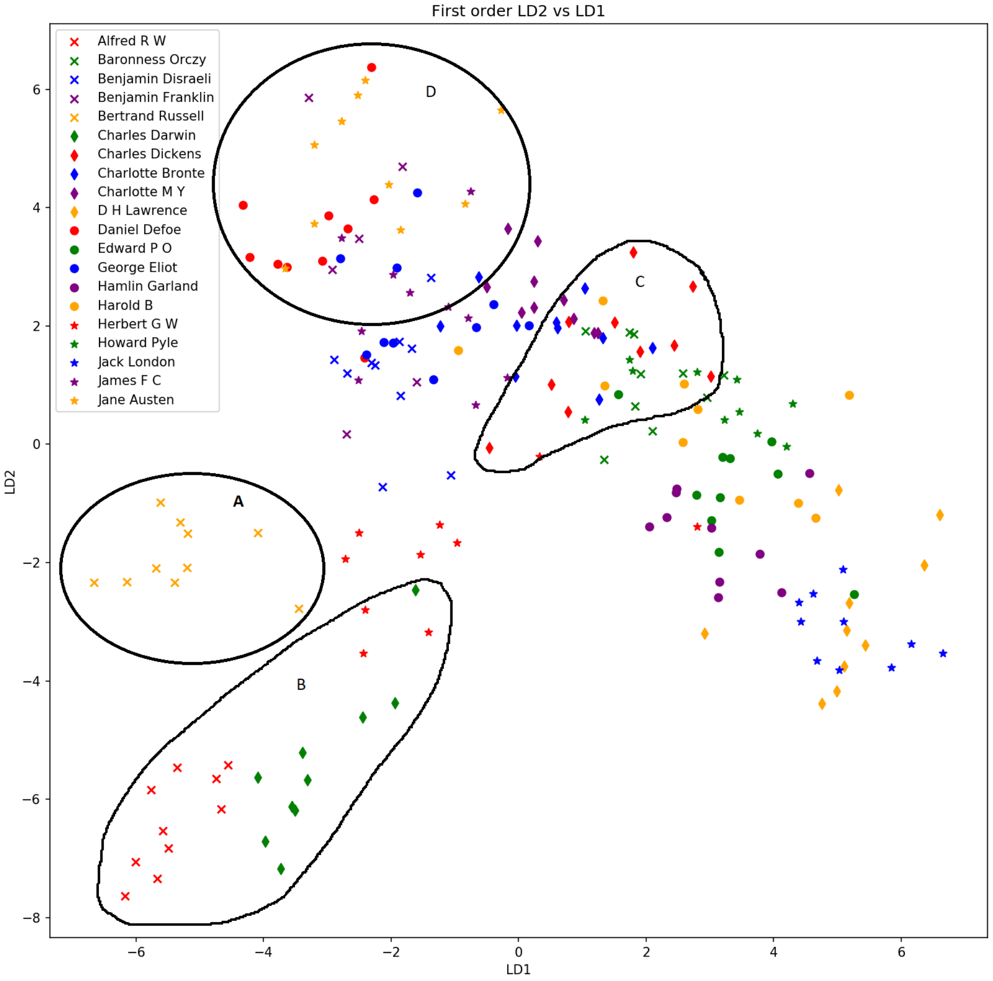


Figure 4.10: LD2 against LD1 scatter plot of 1^{st} order Markov chain prediction model with LDA approach

In **Cluster C**, *Charles Dicken* (Marker: Red diamond) has a moderate intra-class scatter but with poor inter-class scatter. Its centroid has poor separation with others classes in this region resulting in poor precision (0.14) and poor recall (0.12).

In **Cluster D**, *Jane Austen* (Marker: Yellow Star) and *Daniel Defoe* (Marker: Red Circle) has similar intra-class spread and poor inter-class between them, seems to be challenging to classify among these two. Nevertheless, these two author classes have high precision and recall. The reason behind is there are 19 linear discriminant for 20 classes and the scatter plot only shows transformed space of first and second linear discriminant. These two authors are likely to be separable in other linear discriminant space. Similarly, classes that has high inter-class scatter and low intra-class scatter might not be separable in other dimension.

Another factor that affect the classifier performance is the test data. If the writing style in the test data were very different from the training of the same original author, it could be misclassified.

4.2.4 First order Markov Chain Ensemble Model

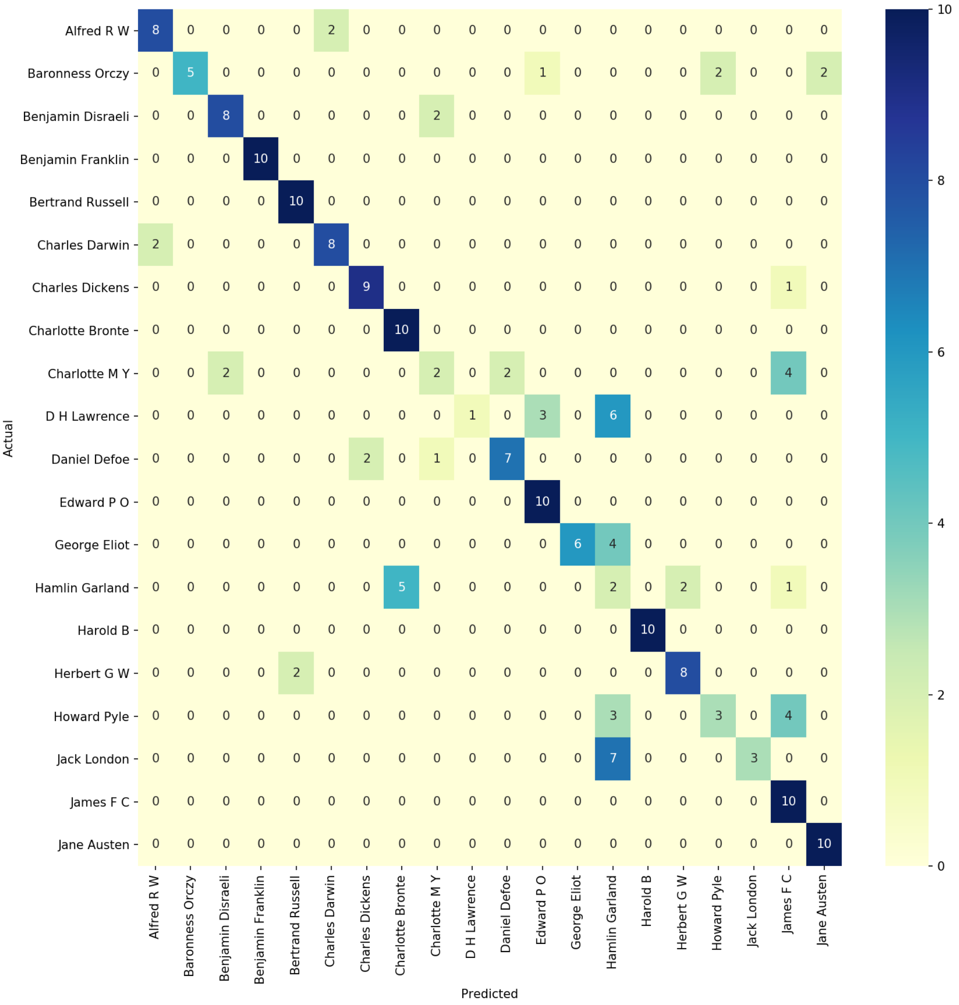


Figure 4.11: Confusion matrix of 1^{st} *order Markov chain ensemble model*

	F1 Score				
	LL	**KLD**	**JSD**	**LDA**	**Ensemble**
Alfred Russel Wallace	**0.82**	0.67	0.69	0.8	0.82
Baronness Orczy	0.67	**0.9**	0.82	0.69	1
Benjamin Disraeli	**0.89**	0.14	0.78	0.53	0.75
Benjamin Franklin	**1**	0.82	0.95	0.95	1
Bertrand Russell	0.87	0.82	0.89	**0.9**	0.95
Charles Darwin	**0.95**	0.67	0.53	0.33	0.78
Charles Dickens	0	0.12	**0.29**	0.17	0.15
Charlotte Bronte	**0.91**	0.53	0.53	0.72	0.91
Charlotte Mary Yonge	0.24	**0.43**	0.15	0.42	0.45
D H Lawrence	0	0	0	**0.64**	0
Daniel Defoe	0.87	0.73	0.78	**0.91**	0.91
Edward Phillips Oppenheim	0.83	0.8	0.83	**0.86**	0.95
George Eliot	**0.82**	0.33	0.33	0.33	0.82
Hamlin Garland	0.07	0.09	0	**0.6**	0.1
Harold Bindloss	**0.91**	0.86	0.74	0.82	0.87
Herbert George Wells	**0.8**	0.78	0.56	0.78	0.82
Howard Pyle	0.29	**0.6**	0.47	0.53	0.62
Jack London	0.5	0.54	**0.8**	0.56	0.82
James Fenimore Cooper	0.48	0.67	**0.77**	0.46	0.62
Jane Austen	0.95	0.7	0.38	**1**	1
Accuracy	0.67	0.58	0.58	0.67	0.74
F1 score	0.64	0.56	0.57	0.65	0.72

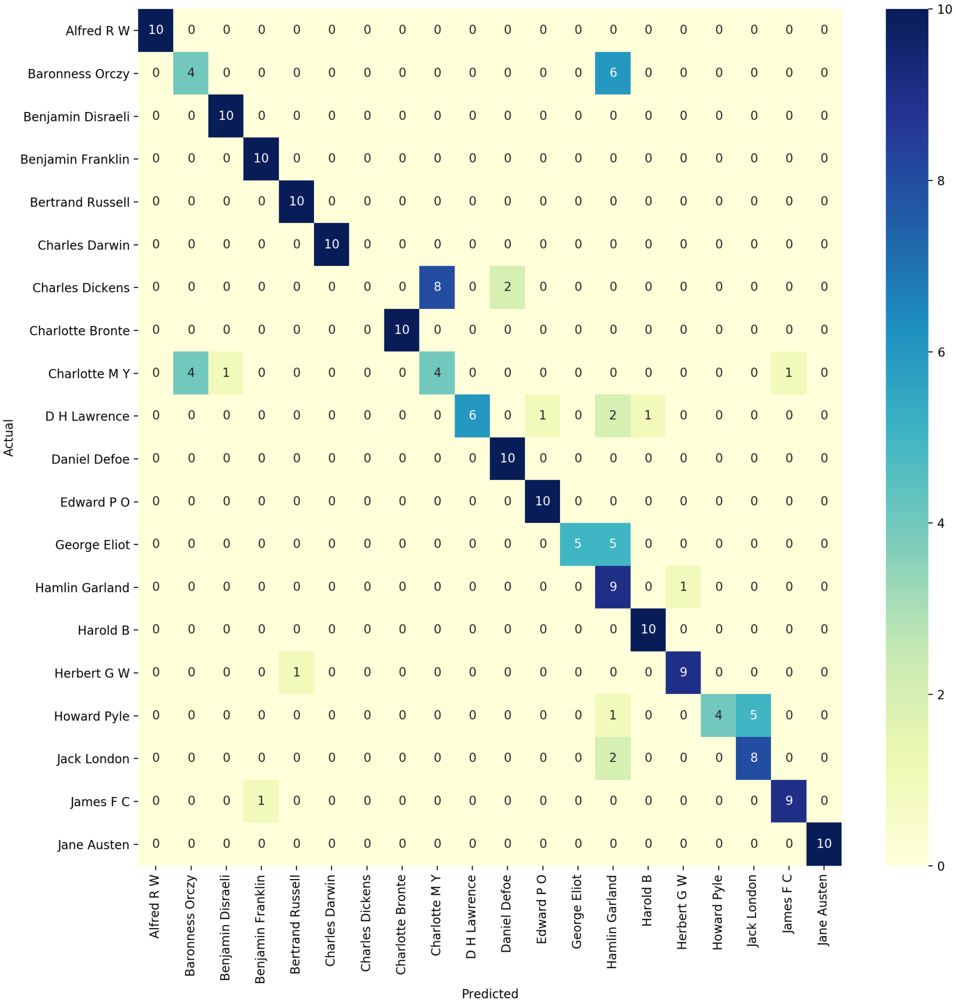
Table 4.5: F1 score of first order Markov chain ensemble model and its member models (best performance member models in bold).

Table 4.5 shows the F1 score of four member models in 1^{st} order MC, noticed that there is no individual member model that dominates the other. Instead, each of them has their own strength on identifying certain classes and complements each other. Thus, the ensemble model four member models (1^{st} order LL, KLD, JSD and LDA model) obtained a higher accuracy of 74% out of the four member models that range between 58% to 67% accuracy.

The early motivation of using ensemble model is because of the use of ensemble can reduce the risk of picking a bad classifier by chance. JSD and KLD model has a poor precision prediction (0.24) in the class Jane Austen but the rest of the model members have almost 100% precision. As a result, the output of the ensemble model of this particular class has increased to 90%. Prediction using ensemble model has successfully increase the performance.

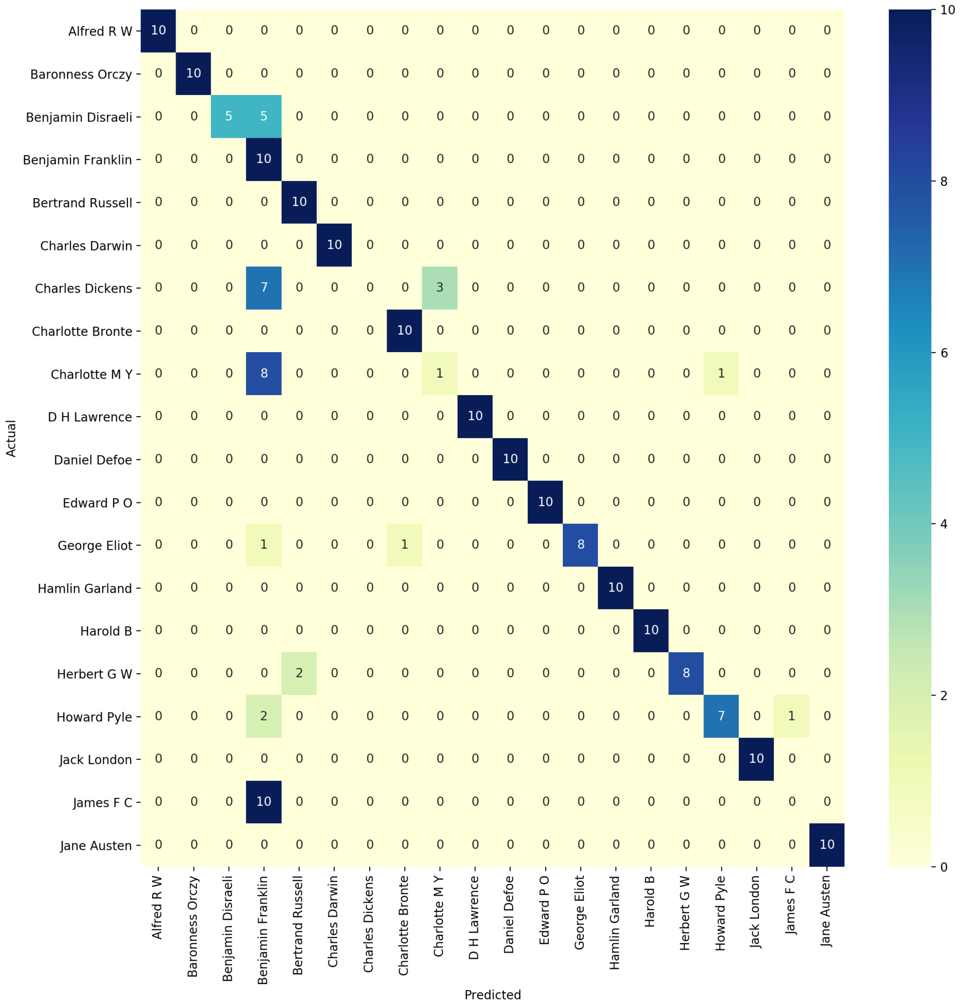
4.3 Higher order Markov Chain

4.3.1 Higher order Markov Chain Log-likelihood Model



	Alfred R W	Baronness Orczy	Benjamin Disraeli	Benjamin Franklin	Bertrand Russell	Charles Darwin	Charles Dickens	Charlotte Bronte	Charlotte M Y	D H Lawrence	Daniel Defoe	Edward P O	George Eliot	Hamlin Garland	Harold B	Herbert G W	Howard Pyle	Jack London	James F C	Jane Austen
Alfred R W	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Baronness Orczy	0	4	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0
Benjamin Disraeli	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Benjamin Franklin	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bertrand Russell	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charles Darwin	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charles Dickens	0	0	0	0	0	0	0	0	8	0	2	0	0	0	0	0	0	0	0	0
Charlotte Bronte	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
Charlotte M Y	0	4	1	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	1	0
D H Lawrence	0	0	0	0	0	0	0	0	0	6	0	1	0	2	1	0	0	0	0	0
Daniel Defoe	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
Edward P O	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
George Eliot	0	0	0	0	0	0	0	0	0	0	0	0	5	5	0	0	0	0	0	0
Hamlin Garland	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	1	0	0	0	0
Harold B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
Herbert G W	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0
Howard Pyle	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4	5	0	0
Jack London	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	8	0	0
James F C	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0
Jane Austen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10

Figure 4.12: Confusion matrix of 2^{nd} order Markov chain prediction model with log-likelihood approach



	Alfred R W	Baronness Orczy	Benjamin Disraeli	Benjamin Franklin	Bertrand Russell	Charles Darwin	Charles Dickens	Charlotte Bronte	Charlotte M Y	D H Lawrence	Daniel Defoe	Edward P O	George Eliot	Hamlin Garland	Harold B	Herbert G W	Howard Pyle	Jack London	James F C	Jane Austen
Alfred R W	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Baronness Orczy	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Benjamin Disraeli	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Benjamin Franklin	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bertrand Russell	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charles Darwin	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charles Dickens	0	0	0	7	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
Charlotte Bronte	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
Charlotte M Y	0	0	0	8	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
D H Lawrence	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0
Daniel Defoe	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
Edward P O	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
George Eliot	0	0	0	1	0	0	0	1	0	0	0	0	8	0	0	0	0	0	0	0
Hamlin Garland	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
Harold B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
Herbert G W	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0
Howard Pyle	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	7	0	1	0
Jack London	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
James F C	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Jane Austen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10

Figure 4.13: Confusion matrix of 3^{rd} order Markov chain prediction model with log-likelihood approach

Figure 4.12 and 4.13 shows the confusion matrix of LL model for 2^{nd} and 3^{rd} order MC. There were more correct prediction made on the diagonal axis of the confusion matrix compare to 1^{st} order MC LL model.

Table 4.6 shows the performance of the 2^{nd} and 3^{rd} order MC LL model. It can be seen that LL model of higher order MC performance greatly increase in every aspect compare to the its 1^{st} order MC. This is because as the increased of Markov chain order, the total number instances in transition matrix increased by factor number of states.

Author	**Second order Markov Chain**			**Third order Markov Chain**		
	Precision	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
Alfred Russel Wallace	1	1	1	1	1	1
Baronness Orczy	0.5	0.4	0.44	1	1	1
Benjamin Disraeli	0.91	1	0.95	1	0.5	0.67
Benjamin Franklin	0.91	1	0.95	0.23	1	0.38
Bertrand Russell	0.91	1	0.95	0.83	1	0.91
Charles Darwin	1	1	1	1	1	1
Charles Dickens	0	0	0	0	0	0
Charlotte Bronte	1	1	1	0.91	1	0.95
Charlotte Mary Yonge	0.33	0.4	0.36	0.25	0.1	0.14
D H Lawrence	1	0.6	0.75	1	1	1
Daniel Defoe	0.83	1	0.91	1	1	1
Edward Phillips Oppenheim	0.91	1	0.95	1	1	1
George Eliot	1	0.5	0.67	1	0.8	0.89
Hamlin Garland	0.36	0.9	0.51	1	1	1
Harold Bindloss	0.91	1	0.95	1	1	1
Herbert George Wells	0.9	0.9	0.9	1	0.8	0.89
Howard Pyle	1	0.4	0.57	0.88	0.7	0.78
Jack London	0.62	0.8	0.7	1	1	1
James Fenimore Cooper	0.9	0.9	0.9	0	0	0
Jane Austen	1	1	1	1	1	1
Accuracy			0.79			0.8
F1 Score	0.8	0.79	0.77	0.8	0.79	0.78

Table 4.6: Second order and third order Markov chain log-likelihood model performance

$$T = n^{m+1}$$

where T is total number of instance in transition matrix, n is the number of states the in first order Markov chain and m is the order of Markov chain. Thus, 1^{st} order MC LL model has $33^2 = 1089$ number of transition probabilities instances. The size of transition matrix of 2^{nd} and 3^{rd} MC increased tremendously, reaching a $33^3 = 35937$ and $33^4 = 1,185,921$ number of transition probabilities.

Higher order MC with more states can improve the authorship attribution task. This is because more features were used in the measuring the likelihood. The condition probabilities of higher MC that uses additional past states as current state refines the probabilities and allow a better prediction. We will demonstrate this by the famous Japanese loanword, “*Sushi*”. Fun fact, ‘‘*Sushi*” the famous Japanese loanword only appeared in Oxford dictionary in year 1893 (Durkin 2014). The English character sequence of “*sus*” followed by “*h*” is exclusive to the word “*sushi*”. Suppose the word ”*Sushi*” appeared in the test set and tested with *Abraham Lincoln* (1809 - 1865).

We are almost certain the text is not written by *Abraham Lincoln* because the loanword only become common after his death. In 3^{rd} order MC, the log-likelihood character sequence of “*sus*” followed by “*h*” is $-\infty$. However, 1^{st} and 2^{nd} order MC cannot rule out the possibility by $-\infty$ log-likelihood because the presence of characters sequences “*su*”, “*us*”, “*sh*”, “*hi*” or “*sus*”, “*ush*”, “*uhi*” in other English words. In another words, the transition probabilities in higher order are more refine and precise which greatly increased the prediction. This are shown in figure 4.14 and 4.15, where we observed the log-likelihood per character of the true author surpass its competitors with increase of MC order. Most importantly is the increase of the gap between the true author and its competitors which indicates reduce of ambiguity and better prediction.

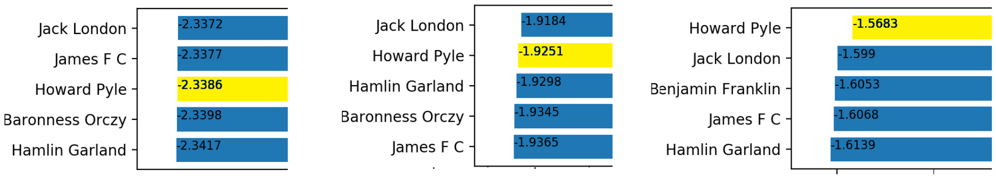


Figure 4.14: Log-likelihood per character ranking, starting from left: 1^{st} *MC,* 2^{nd} *MC,* 3^{rd} *MC . Ground truth author (Howard Pyle) highlighted in yellow*

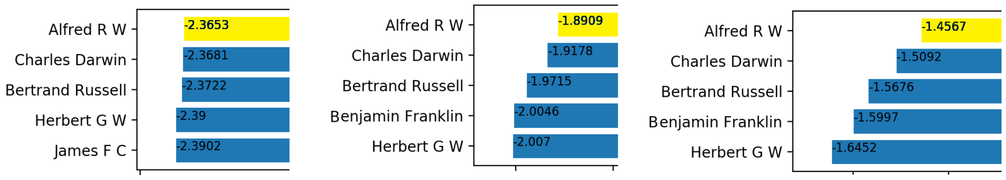


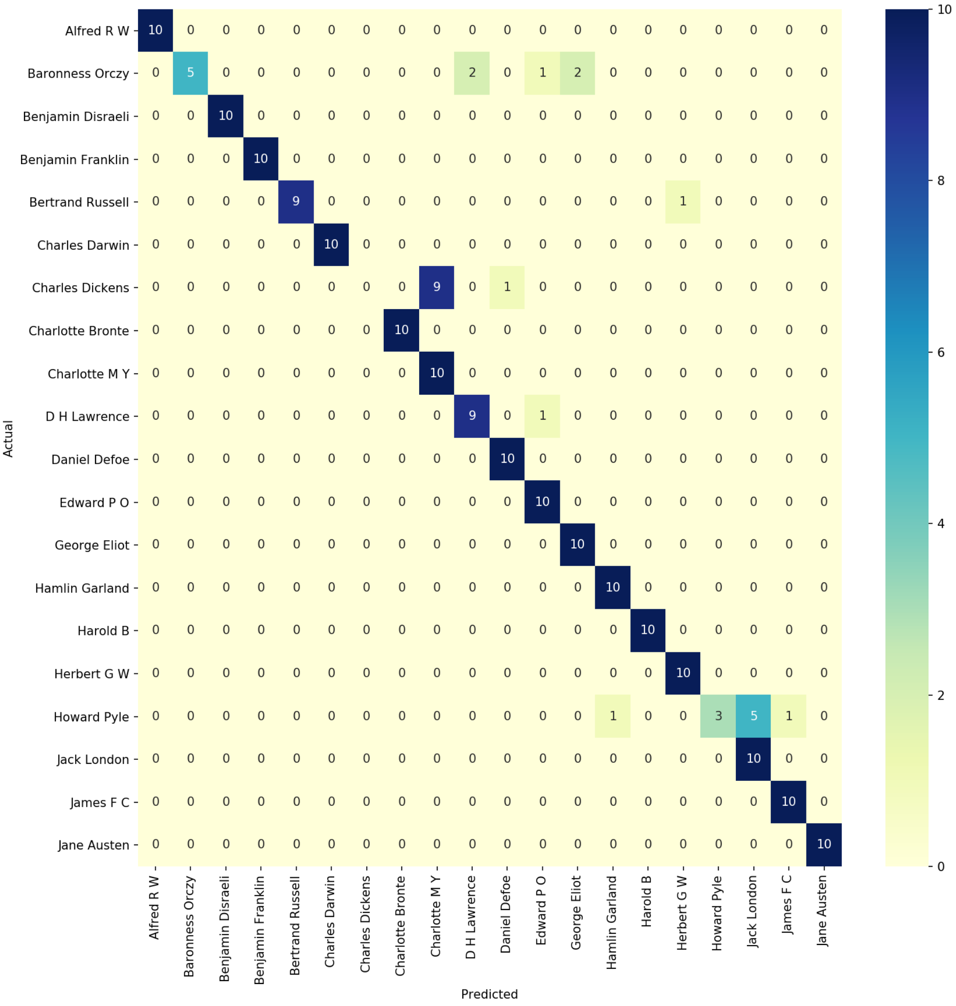
Figure 4.15: Log-likelihood per character ranking, starting from left: 1^{st} *MC,* 2^{nd} *MC,* 3^{rd} *MC . Ground truth author (Alfred R W) highlighted in yellow*

In 1^{st} order MC LL model, figure 4.4 shows a systematic misclassification of the authorship to *James F C* and *Hamlin Garland*. Although the systematic misclassification to *Hamlin Garland* was inherit in 2^{nd} order MC LL model in figure 4.12, 3^{rd} order MC LL model chain gives perfect performance (recall, precision and F1-score of 1) in classifying this particular class.

Besides, we also observed fluctuation of performance in identifying certain in different MC order. For instance, *James F C* has moderate F1 score (0.48) at 1^{st} order MC, excellent F1 score (0.9) at 2^{nd} order MC but dropped to a zero F1-score at 3^{rd} order. This is because different features were captured at different order. This also open the opportunity to built an ensemble model of different Markov chain order.

We are anticipated to see the experiment result above 3^{rd} order MC because the transition matrix starts to encodes information between two words separated by a blank space. Unfortunately, due to limitation of memory and computing hardware, the authorship attribution task ends here. We foresee the performance of the LL model will continue improve with increase of MC order, until the transition probabilities were too specific and no longer can be used as a stylometry feature. In addition, higher order Markov chain associate with higher dimensionality, imagine our Markov chain with 33 characters at 3^{rd} order MC has $33^4 \approx 1M$, 4^{th} order MC $33^5 \approx 39M$. The size of our training data (320,000 to 48,000,000 characters) is inadequate to record the occurrence of character sequences and turn into a reliable transition probabilities that reflects the true stylometry features. Moreover, from the practicality perspective, it is infeasible in real world to obtain sufficient written sample of an author.

4.3.2 Higher order Markov Chain KLD Model



	Alfred R W	Baronness Orczy	Benjamin Disraeli	Benjamin Franklin	Bertrand Russell	Charles Darwin	Charles Dickens	Charlotte Bronte	Charlotte M Y	D H Lawrence	Daniel Defoe	Edward P O	George Eliot	Hamlin Garland	Harold B	Herbert G W	Howard Pyle	Jack London	James F C	Jane Austen
Alfred R W	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Baronness Orczy	0	5	0	0	0	0	0	0	0	2	0	1	2	0	0	0	0	0	0	0
Benjamin Disraeli	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Benjamin Franklin	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bertrand Russell	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Charles Darwin	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charles Dickens	0	0	0	0	0	0	0	0	9	0	1	0	0	0	0	0	0	0	0	0
Charlotte Bronte	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
Charlotte M Y	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0
D H Lawrence	0	0	0	0	0	0	0	0	0	9	0	1	0	0	0	0	0	0	0	0
Daniel Defoe	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
Edward P O	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
George Eliot	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0
Hamlin Garland	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
Harold B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
Herbert G W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0
Howard Pyle	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	3	5	1	0
Jack London	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
James F C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0
Jane Austen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10

Figure 4.16: Confusion matrix of 2^{nd} order Markov chain prediction model with KL divergence approach

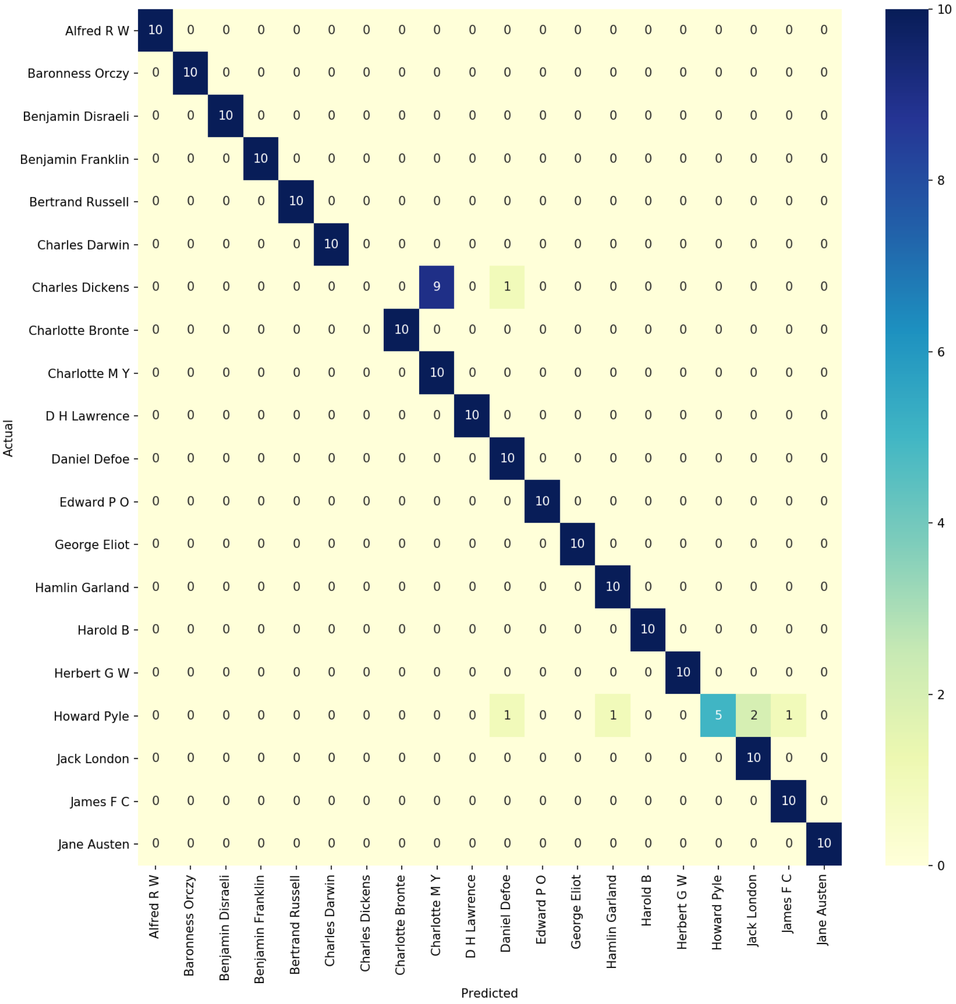


Figure 4.17: Confusion matrix of 3^{rd} order Markov chain prediction model with KL approach

Figure 4.16 and 4.17 shows the confusion matrix of KLD model for 2^{nd} and 3^{rd} order MC using $\mathbf{D_{KL\ q(x)}}$ value of 1.5 and 1.9. The model had achieved 88% accuracy and 93% while baseline result with random guess is 5%.

Earlier, KLD model of first order Markov chain obtained a low accuracy (58%) and low F1 score (0.56). Table 4.7 shows the class performance of the 2^{nd} and 3^{rd} order MC KLD model. It can be seen that KLD model of higher order MC performance greatly increase in every aspect compare to the 1^{st} order MC KLD model in table 4.2. KLD model at 2^{nd} order MC has an high accuracy (86%) and F1 score (0.83) and further improved in 3^{rd} order MC with an impressive accuracy (93%) and F1 score (0.9). This is because more stylometry features were used at authorship attribution at higher order MC and further refine the difference between the writing

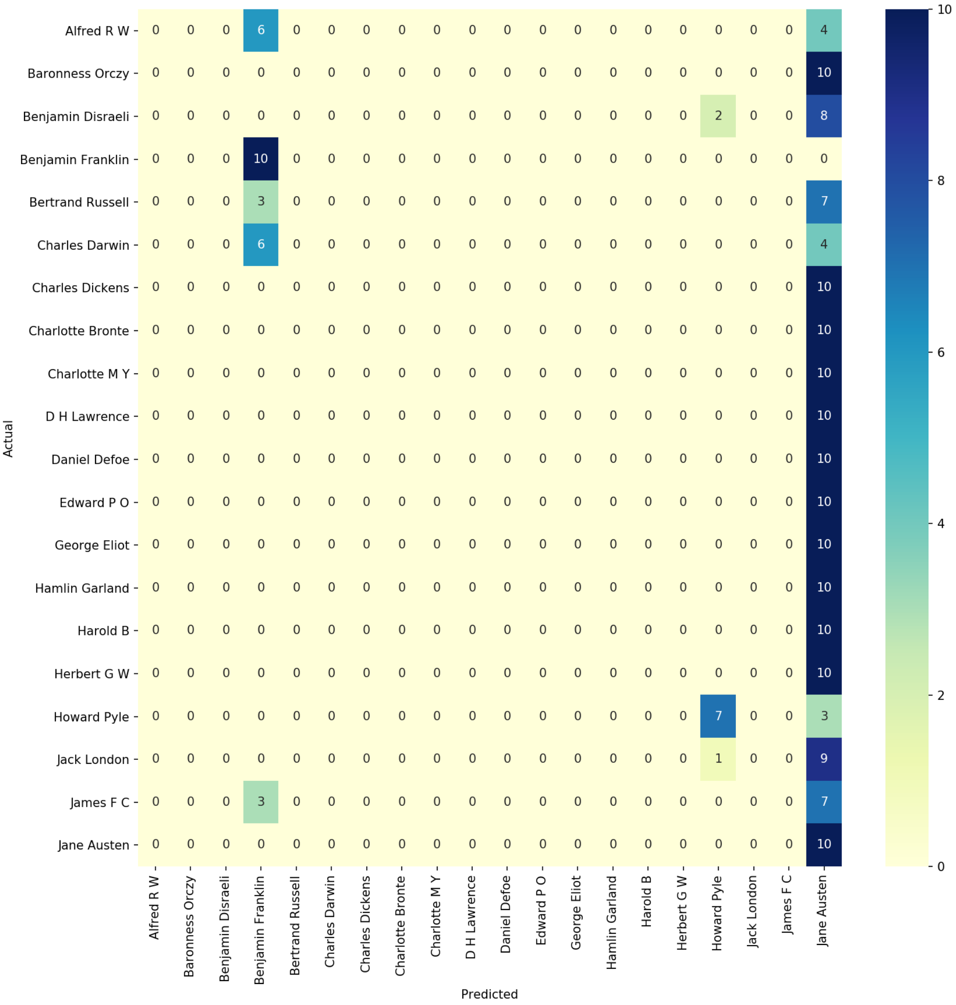
Author	**Second order Markov Chain**			**Third order Markov Chain**		
	Precision	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
Alfred Russel Wallace	1	1	1	1	1	1
Baronness Orczy	1	0.4	0.57	1	1	1
Benjamin Disraeli	0.91	1	0.95	1	1	1
Benjamin Franklin	1	1	1	1	1	1
Bertrand Russell	1	1	1	1	1	1
Charles Darwin	1	1	1	1	1	1
Charles Dickens	0	0	0	0	0	0
Charlotte Bronte	0.83	1	0.91	1	1	1
Charlotte Mary Yonge	0.44	0.7	0.54	0.53	1	0.69
D H Lawrence	0.77	1	0.87	1	1	1
Daniel Defoe	0.77	1	0.87	0.83	1	0.91
Edward Phillips Oppenheim	0.91	1	0.95	1	1	1
George Eliot	1	1	1	1	1	1
Hamlin Garland	0.77	1	0.87	0.91	1	0.95
Harold Bindloss	1	1	1	1	1	1
Herbert George Wells	1	1	1	1	1	1
Howard Pyle	1	0.3	0.46	1	0.5	0.67
Jack London	0.67	1	0.8	0.83	1	0.91
James Fenimore Cooper	0.89	0.8	0.84	0.91	1	0.95
Jane Austen	1	1	1	1	1	1
Accuracy			0.86			0.93
F1 Score	0.85	0.86	0.83	0.9	0.93	0.9

Table 4.7: Second order and third order Markov chain KL divergence model performance table

style of the authors.

From table 4.7, we can see neither 2^{nd} order nor 3^{nd} order MC KLD model is completely better than another. For instance, the former is a better classifier for author class *Benjamin Franklin*, *Charlotte marry Yonge* and *James Fenimore* while the latter are better in classifying author class *Baronness Orczy* and *Howard Pyle*. This is because different Markov chain order with different size of transition matrices encodes different stylometry features. As discussed before section 4.3.1, this is because as the increased of Markov chain order, the number of transition probabilities greatly increase. Thus, the transition matrix can extract more stylometry features and further refines the difference between the true author and its competitors. The comparison of the KLD and JSD model will be discussed in the next section.

4.3.3 Higher order Markov Chain JSD Model



	Alfred R W	Baronness Orczy	Benjamin Disraeli	Benjamin Franklin	Bertrand Russell	Charles Darwin	Charles Dickens	Charlotte Bronte	Charlotte M Y	D H Lawrence	Daniel Defoe	Edward P O	George Eliot	Hamlin Garland	Harold B	Herbert G W	Howard Pyle	Jack London	James F C	Jane Austen
Alfred R W	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
Baronness Orczy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
Benjamin Disraeli	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	8
Benjamin Franklin	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bertrand Russell	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
Charles Darwin	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
Charles Dickens	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
Charlotte Bronte	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
Charlotte M Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
D H Lawrence	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
Daniel Defoe	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
Edward P O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
George Eliot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
Hamlin Garland	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
Harold B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
Herbert G W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
Howard Pyle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	3
Jack London	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	9
James F C	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
Jane Austen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10

Figure 4.18: Confusion matrix of 2^{nd} order Markov chain prediction model with JS divergence approach

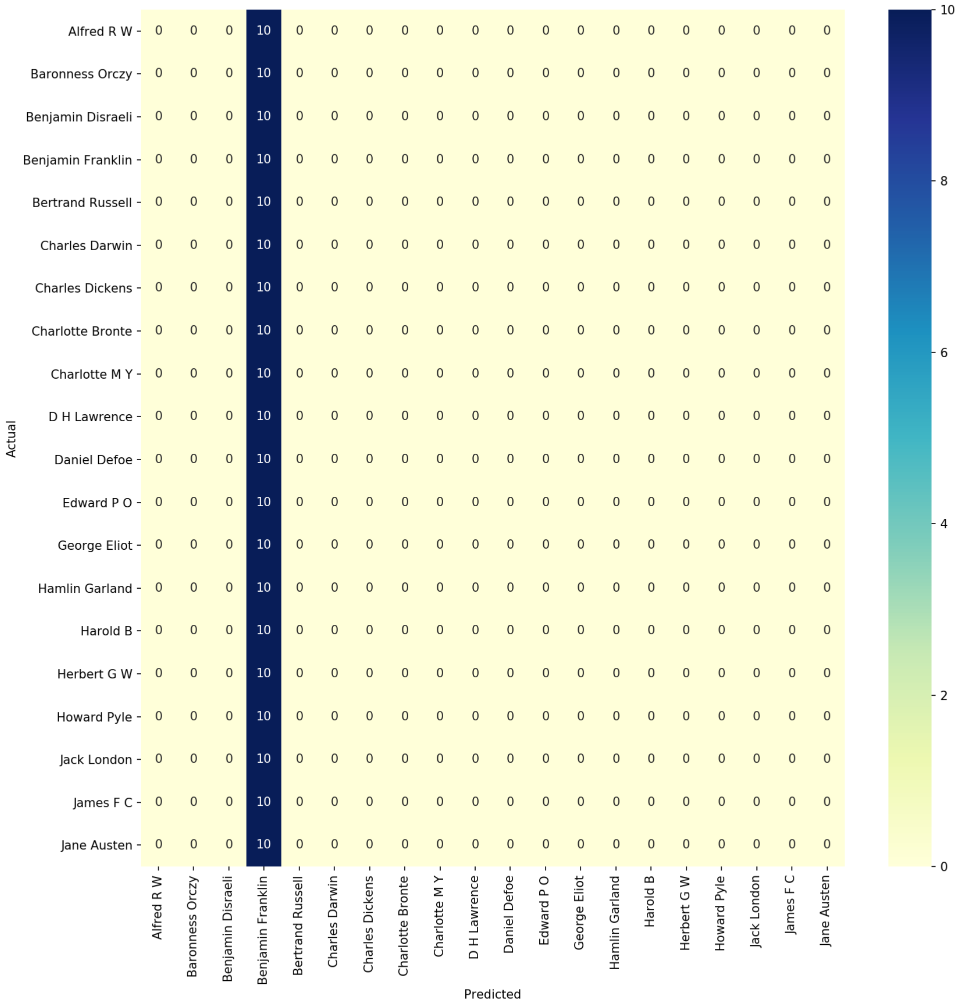


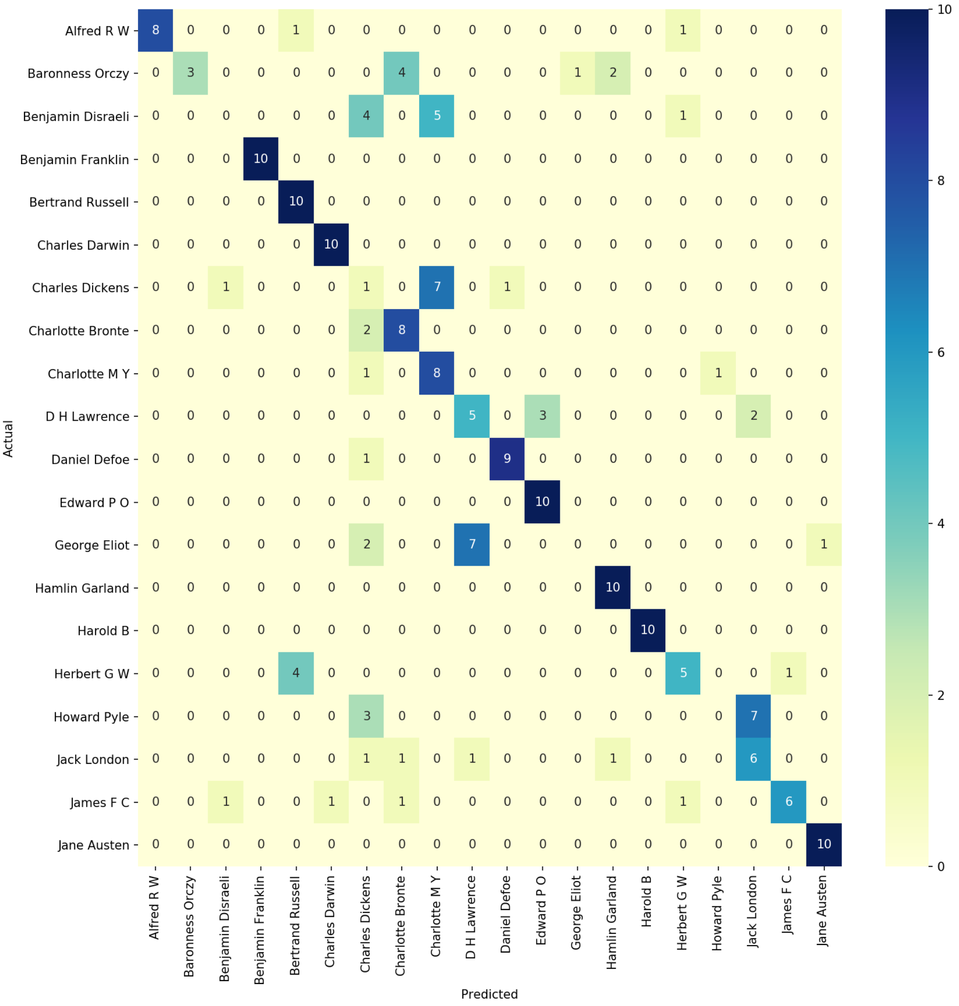
Figure 4.19: Confusion matrix of 3^{rd} order Markov chain prediction model with JS approach

Figure 4.18 and 4.19 shows the confusion matrix of JSD model for 2^{nd} order and 3^{rd} order MC. Clearly, the JSD model failed to classify the text in 2^{nd} and 3^{rd} order MC. The culprit behind the breakdown is the zero-probability event. In 1^{st} order MC, there is very little zero-value instances in the transition matrix. As the size of the transition matrix grows with the order Markov rhain, there are more zeros in both training and testing transition matrices.

Let us recall that KLD measurement is undefined when there when one of the test distribution $p(x)$ or train transition matrix distribution $q(x)$ is zero. The assumption to apply KLD is $p(x)$ distribution is a subset of $q(x)$ distribution, that is, it only works when $p(X)$ and $q(x)$ is non-zero. KLD uses the equation $p(x)ln\frac{p(x)}{q(x)}$ to measure how many additional computer bits (for log_2) when a distribution $q(x)$ is used to approximate another true distribution $p(x)$. Thus, when non-zero $q(x)$ is used to approximate $p(x) = 0$, that is fine because it means the system is wasting bits to encode. However, all the bits in the universe would not be enough to encode when $q(x) = 0$ is used to encode a non-zero distribution of $p(x) > 0$.

This limitation was inherit to JSD measurement. There are few solutions to avoid such case. The first way was applied in our KLD model, that is to make assumption for such case. For instance, we defined $\mathbf{D_{KL\ q(x)}}$ value for $p(x) = 0, q(x) > 0$ scenario to work. The next workaround is to smooth distribution by applying a non-zero prior distribution such as the characters distribution in table 2.1. Another alternative is to allocate a count of one in the transition matrix before sampling the occurrence of character sequence, by this we assumed occurrence of any character sequence is possible from typing error.

4.3.4 Higher order LDA Model



	Alfred R W	Baronness Orczy	Benjamin Disraeli	Benjamin Franklin	Bertrand Russell	Charles Darwin	Charles Dickens	Charlotte Bronte	Charlotte M Y	D H Lawrence	Daniel Defoe	Edward P O	George Eliot	Hamlin Garland	Harold B	Herbert G W	Howard Pyle	Jack London	James F C	Jane Austen
Alfred R W	8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Baronness Orczy	0	3	0	0	0	0	0	4	0	0	0	0	1	2	0	0	0	0	0	0
Benjamin Disraeli	0	0	0	0	0	0	4	0	5	0	0	0	0	0	0	1	0	0	0	0
Benjamin Franklin	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bertrand Russell	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charles Darwin	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charles Dickens	0	0	1	0	0	0	1	0	7	0	1	0	0	0	0	0	0	0	0	0
Charlotte Bronte	0	0	0	0	0	0	2	8	0	0	0	0	0	0	0	0	0	0	0	0
Charlotte M Y	0	0	0	0	0	0	1	0	8	0	0	0	0	0	0	0	1	0	0	0
D H Lawrence	0	0	0	0	0	0	0	0	0	5	0	3	0	0	0	0	0	2	0	0
Daniel Defoe	0	0	0	0	0	0	1	0	0	0	9	0	0	0	0	0	0	0	0	0
Edward P O	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
George Eliot	0	0	0	0	0	0	2	0	0	7	0	0	0	0	0	0	0	0	0	1
Hamlin Garland	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
Harold B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
Herbert G W	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	5	0	0	1	0
Howard Pyle	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	7	0	0
Jack London	0	0	0	0	0	0	1	1	0	1	0	0	0	1	0	0	0	6	0	0
James F C	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	6	0
Jane Austen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10

Figure 4.20: Confusion matrix of 2^{nd} order Markov chain prediction model with LDA approach

Author	**Second order Markov Chain**		
	Precision	**Recall**	**F1 Score**
Alfred Russel Wallace	1	0.8	0.89
Baronness Orczy	1	0.3	0.46
Benjamin Disraeli	0	0	0
Benjamin Franklin	1	1	1
Bertrand Russell	0.67	1	0.8
Charles Darwin	0.91	1	0.95
Charles Dickens	0.07	0.1	0.08
Charlotte Bronte	0.57	0.8	0.67
Charlotte Mary Yonge	0.4	0.8	0.53
D H Lawrence	0.38	0.5	0.43
Daniel Defoe	0.9	0.9	0.9
Edward Phillips Oppenheim	0.77	1	0.87
George Eliot	0	0	0
Hamlin Garland	0.77	1	0.87
Harold Bindloss	1	1	1
Herbert George Wells	0.62	0.5	0.56
Howard Pyle	0	0	0
Jack London	0.4	0.6	0.48
James Fenimore Cooper	0.86	0.6	0.71
Jane Austen	0.91	1	0.95
Accuracy			0.65
F1 Score	0.61	0.65	0.61

Table 4.8: Second order Markov chain LDA model performance table

Figure 4.20 and table 4.8 shows the confusion matrix and the performance of 2^{nd} order MC LDA model. The model obtained accuracy of 65% and F1 score of 0.61. It is disappointing that its performance had slightly dropped from its 1^{st} order MC .

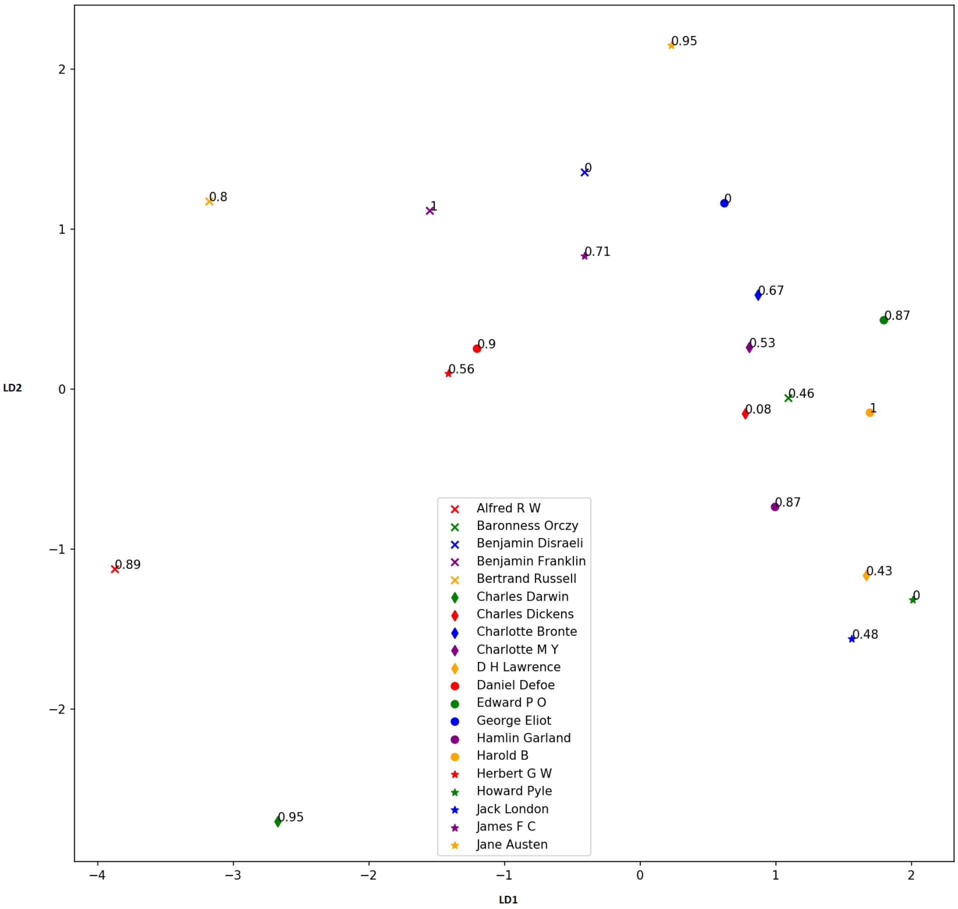


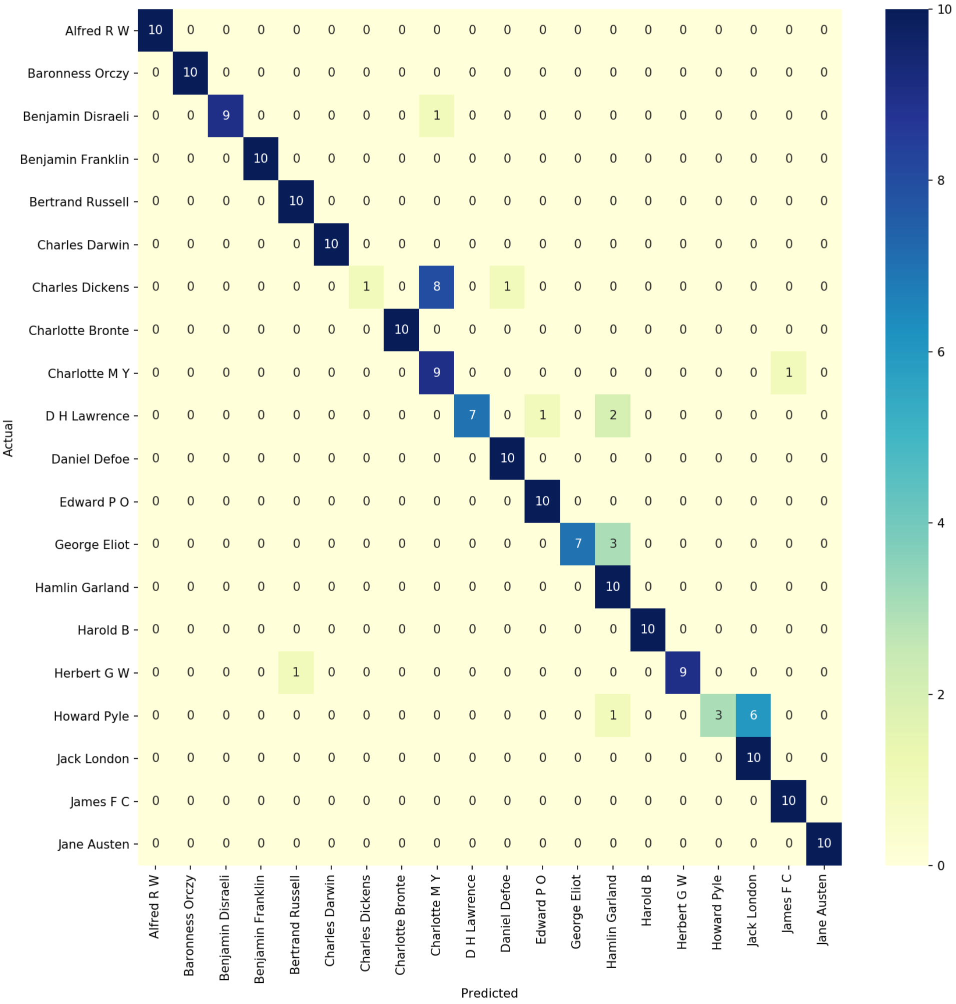
Figure 4.21: Transformed training data by mean in LD2 against LD1 scatter plot of 2^{nd} order Markov chain prediction model with LDA approach, Labelled in F1 score

Figure 4.21 shows the centroid of training data in transformed space, LD2 against LD1 (labelled with class F1 score). It can be seen that the author class with high F1 score associate with large centroid separation with other classes. In LDA classification, transformed testing data are assigned to the class of nearest centroid. Classes that has short centroid distance are likely to be have low precision and low recall. This is because the transformed testing data were usually scattered around the centroid of the training data. The testing data might be misclassified if the nearest centroid was not the true author. A large separation between the centroid is favourable in classification.

The reason LDA model did not improve with higher order MC is because LDA assumed normally distributed and not equal variance in all the attribute, which is not true. Especially in higher order MC, the character sequences were specific and the row distribution in the transition matrix limit only on a few characters. As the transition probabilities distribution of higher order MC could be extreme at both range [0,1], the mean of each class does not provide a good reference point to be maximised in the equation 3.11. Lastly, LDA analysis is beyond 2^{nd} order MC is halted due to the memory and hardware limitation of the computer.

4.4 Project Ensemble model

The project ensemble model the combination of all authorship attribution model in this project excluding 2^{nd} and 3^{rd} order MC JSD model. Figure 4.22 presents the confusion matrix of the model. The combination of all the models in this projects uses different stylometry features from three different order of MC, seeking to produce a balance and reliable prediction. From the confusion matrix, the model had achieved great performance. There were many correctly identified classes on the diagonal axis of the confusion matrix. However, like its member models, the models experience difficulty in identifying the author class of *Charles Dickens* and *Howard Pyle*, which is likely the writing style of the test set happen to be different from other work. Table 4.9 presents the performance table by class of the ensemble model. The performance of ensemble model using all models (except higher order JSD model) achieved a 88$\%$ accuracy and 0.86 F1 score, a performance which is higher than most of the member models (Except 3^{rd} order MC KLD model).



	Alfred R W	Baronness Orczy	Benjamin Disraeli	Benjamin Franklin	Bertrand Russell	Charles Darwin	Charles Dickens	Charlotte Bronte	Charlotte M Y	D H Lawrence	Daniel Defoe	Edward P O	George Eliot	Hamlin Garland	Harold B	Herbert G W	Howard Pyle	Jack London	James F C	Jane Austen
Alfred R W	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Baronness Orczy	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Benjamin Disraeli	0	0	9	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Benjamin Franklin	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bertrand Russell	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charles Darwin	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charles Dickens	0	0	0	0	0	0	1	0	8	0	1	0	0	0	0	0	0	0	0	0
Charlotte Bronte	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
Charlotte M Y	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	1	0
D H Lawrence	0	0	0	0	0	0	0	0	0	7	0	1	0	2	0	0	0	0	0	0
Daniel Defoe	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
Edward P O	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
George Eliot	0	0	0	0	0	0	0	0	0	0	0	0	7	3	0	0	0	0	0	0
Hamlin Garland	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
Harold B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
Herbert G W	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0
Howard Pyle	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	3	6	0	0
Jack London	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
James F C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0
Jane Austen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10

Figure 4.22: Confusion matrix of ensemble model using combination of 1^{st} order MC LL, KLD, JSD, LDA model, 2^{nd} order MC LL, KLD, LDA model and $3rd$ order MC LL KLD model

	Precision	**Recall**	**F1 Score**
Alfred Russel Wallace	1	1	1
Baronness Orczy	1	1	1
Benjamin Disraeli	1	0.9	0.95
Benjamin Franklin	1	1	1
Bertrand Russell	0.91	1	0.95
Charles Darwin	1	1	1
Charles Dickens	1	0.1	0.18
Charlotte Bronte	1	1	1
Charlotte Mary Yonge	0.5	0.9	0.64
D H Lawrence	1	0.7	0.82
Daniel Defoe	0.91	1	0.95
Edward Phillips Oppenheim	0.91	1	0.95
George Eliot	1	0.7	0.82
Hamlin Garland	0.62	1	0.77
Harold Bindloss	1	1	1
Herbert George Wells	1	0.9	0.95
Howard Pyle	1	0.3	0.46
Jack London	0.62	1	0.77
James Fenimore Cooper	0.91	1	0.95
Jane Austen	1	1	1
Accuracy			0.88
F1 score	0.92	0.88	0.86

Table 4.9: Performance table of ensemble model using combination of 1^{st} *order MC LL, KLD, JSD, LDA,* 2^{nd} *order MC LL, KLD, LDA model and* $3rd$ *order MC LL KLD model*

4.5 Summary

So far, different authorship attribution models such as LL, KLD, JSD, LDA and their ensemble models were evaluated. In 1^{st} order Markov chain, LL and LDA model has the highest performance followed by KLD and JSD model. The ensemble model with these combinations were able to produce a higher performance than its member model. 1^{st} Markov chain uses the probability of current state of single character and next state of single character as the stylometry features to distinguish between authors. The character features extracted were low level features that only extract the information between two characters. As a result, the best model 1^{st} order MC model, i.e. the 1^{st} order ensemble model obtained an accuracy of 74% accuracy and F1 score of 0.72. Nevertheless, it has the advantages of fewer little zero-probability instances in the transition matrix. This is desirable for models like LL, KLD and JSD model because the equation of these models requires further definition in zero probability event.

In 2^{nd} order MC, KLD model further improved to an accuracy of 88% and F1 score of 0.85. Its 3^{rd} order MC is the best performance model in the project gave the an accuracy of 93% and F1 score of 0.9 . JSD model breakdown in higher order MC due to the abundance of zero-probability instances in the transition matrix. Although LL model obtained a slightly better result in 2^{nd} order MC, its performance did not improve at 3^{rd} order MC. On the other hand, 2^{nd} order MC LDA model has a slightly lower performance than 1^{st} order MC. In general, the authorship attribution model improved in higher order MC as the more transition probabilities as stylometry features were used and the probabilities were more refined. The performance of ensemble model using all models (excluding higher order JSD model) achieved an accuracy of 88% accuracy and F1 score of 0.86, a performance which is higher than most of the member models (Except 3^{rd} order MC KLD model). Therefore, we recommend this ensemble model to be used in authorship attribution task for its high accuracy and high F1 score. However, if the user are concerned with its long run time, running with only 1^{st} order MC models were able to produce significant result. Besides, the user are also able to select a model that fulfil their specification. For instance, selecting 3^{rd} order MC LL model (Class *Daniel Defoe* - Precision:1, Recall: 1, F1 score :1) to identify the author class *Daniel Defoe* instead of using ensemble model (Class *Daniel Defoe* - Precision:0.91, Recall: 1, F1 score :0.95).

Zero-probability instances in the transition matrix can be an issue in authorship attribution. We proposed few methods including assigning a value for KLD method, taking the log-likelihood per character, or assign a non-zero prior distribution. We also discussed the LDA model performance related with the centroid distance of the class in the transformed space.

Chapter 5

Conclusion

This study had developed various authorship attribution model using Markov chain at different order with the assumption that the writing style of the author were unique and consistent in all his work. The study illustrates that characters-level features modelled by Markov chain is an effective features that discriminate the true author from its competitors. The CRISP-DM framework was implemented in the development of the authorship attribution model, covering business understanding to evaluation phase.

In data understanding phase, data quality issues such as minority characters, consecutive blank spaces and newlines were identified with examples from the data set. Hence, necessary data cleaning procedures such character replace, grouping and blank space merging were taken to address the data quality issues. The stylometry features of first order to third order Markov chain were extracted from the training data and testing data into the transition matrix. Four methods including log-likelihood, Kullback–Leibler divergence, Jensen-Shannon Divergence and Linear Discriminant analysis were proposed to measure similarity of writing style between the given word and the written sample of the authors. Later, the prediction by various method at different order were integrated with ensemble model by majority voting. Ensemble model with majority voting were proven to deliver a better prediction.

In general, higher performance were obtained using higher order Markov chain. As the size of the transition matrix grows, the transition probabilities were more specific and increased in number. The third order Markov chain KLD model gave the highest performance in the authorship attribution task.

Bibliography

Argamon, S., Koppel, M., Fine, J. & Shimoni, A. R. (2003), 'Gender, genre, and writing style in formal written texts', *Text & Talk* **23**(3), 321–346.

Atangana, A. & Gómez-Aguilar, J. (2018), 'Fractional derivatives with no-index law property: application to chaos and statistics', *Chaos, Solitons & Fractals* **114**, 516–535.

Bellman, R. (1966), 'Dynamic programming', *Science* **153**(3731), 34–37.

Cover, T. M. & Thomas, J. A. (2006), 'Elements of information theory', p. 19.

Durkin, P. (2014), 'Where do our words come from?', *The Guardian* .
URL: *https://www.theguardian.com/education/2014/feb/07/quiz-etymology-word-origins-answers*

Fawcett, T. & Provost, F. J. (1996), Combining data mining and machine learning for effective user profiling., *in* 'KDD', pp. 8–13.

Finkenstaedt, T. & Wolff, D. (1973), *Ordered profusion; studies in dictionaries and the English lexicon*, Vol. 13, C. Winter.

Fisher, R. A. (1936), 'The use of multiple measurements in taxonomic problems', *Annals of eugenics* **7**(2), 179–188.

Gao, X., Ren, B., Zhang, H., Sun, B., Li, J., Xu, J., He, Y. & Li, K. (2020), 'An ensemble imbalanced classification method based on model dynamic selection driven by data partition hybrid sampling', *Expert Systems with Applications* **160**, 113660.
URL: *http://www.sciencedirect.com/science/article/pii/S095741742030484X*

Guo, Y., Hastie, T. & Tibshirani, R. (2007), 'Regularized linear discriminant analysis and its application in microarrays', *Biostatistics* **8**(1), 86–100.

Han, J., Pei, J. & Kamber, M. (2011), *Data mining: concepts and techniques*, Elsevier.

Hoover, D. L. (2004), 'Testing burrows's delta', *Literary and linguistic computing* **19**(4), 453–475.

Kay, G. (1995), 'English loanwords in japanese', *World Englishes* **14**(1), 67–76.

Kotu, V. & Deshpande, B. (2015), Chapter 2 - data mining process, *in* V. Kotu & B. Deshpande, eds, 'Predictive Analytics and Data Mining', Morgan Kaufmann, Boston, pp. 17 – 36.
URL: *http://www.sciencedirect.com/science/article/pii/B9780128014608000021*

Lahiri, S. (2014), Complexity of Word Collocation Networks: A Preliminary Structural Analysis, *in* 'Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics', Association for Computational Linguistics, Gothenburg, Sweden, pp. 96–105.
URL: *http://www.aclweb.org/anthology/E14-3011*

Mandenhal, T. (1887), 'The characteristics curves of composition science', *Science* **9**(214s), 237–46.

Morozov, N. (1915), 'Lingvisticheskie spektry (linguistic spectrums)', *Izvestia Akademii Nauk* pp. 1–4.

Mosteller, F. & Wallace, D. L. (1964), *The federalist: Inference and disputed authorship*, Addison-Wesley.

Neal, T., Sundararajan, K., Fatima, A., Yan, Y., Xiang, Y. & Woodard, D. (2017), 'Surveying stylometry techniques and applications', *ACM Computing Surveys (CSUR)* **50**(6), 1–36.

Niculescu-Mizil, A., Perlich, C., Swirszcz, G., Sindhwani, V., Liu, Y., Melville, P., Wang, D., Xiao, J., Hu, J., Singh, M., Shang, W. & Zhu, Y. (2009), 'Winning the kdd cup orange challenge with ensemble selection.', *Journal of Machine Learning Research - Proceedings Track* **7**, 23–34.

Polikar, R. (2006), 'Ensemble based systems in decision making', *IEEE Circuits and Systems Magazine* **6**(3), 21–45.

Project Gutenberg (2018). `https://www.gutenberg.org`, last accessed on 27/07/20.
URL: *http://www.gutenberg.org/*

Qian, Y., Liang, Y., Li, M., Feng, G. & Shi, X. (2014), 'A resampling ensemble algorithm for classification of imbalance problems', *Neurocomputing* **143**, 57 – 67.
URL: *http://www.sciencedirect.com/science/article/pii/S0925231214007644*

Ramnial, H., Panchoo, S. & Pudaruth, S. (2016), Authorship attribution using stylometry and machine learning techniques, *in* 'Intelligent Systems Technologies and Applications', Springer, pp. 113–125.

Rao, C. R. (1948), 'The utilization of multiple measurements in problems of biological classification', *Journal of the Royal Statistical Society. Series B (Methodological)* **10**(2), 159–203.

Rastogi, V. & Shivam, P. M. (2018), Authorship detection in cyber world, *in* '3rd National Conference on Image Processing, Computing, Communication, Networking and Data Analytics', p. 318.

Richardson, M., Gabrosek, J., Reischman, D. & Curtiss, P. (2004), 'Morse code, scrabble, and the alphabet', *Journal of Statistics Education* **12**(3).

Rocha, A., Scheirer, W. J., Forstall, C. W., Cavalcante, T., Theophilo, A., Shen, B., Carvalho, A. R. B. & Stamatatos, E. (2017), 'Authorship attribution for social media forensics', *IEEE Transactions on Information Forensics and Security* **12**(1), 5–33.

Shearer, C. (2000), 'The CRISP-DM model: the new blueprint for data mining', *Journal of data warehousing* **5**(4), 13–22.

Stamatatos, E. (2009), 'A survey of modern authorship attribution methods', *Journal of the American Society for information Science and Technology* **60**(3), 538–556.

Suman Patil, S. V. N. K. Y. (2019), 'Email authorship attribution'.

Tennyson, M. F. (2012), On improving authorship attribution of source code, *in* 'International Conference on Digital Forensics and Cyber Crime', Springer, pp. 58–65.

Töscher, A. & Jahrer, M. (2009), 'The bigchaos solution to the netflix grand prize'.

Von Hilgers, P. & Langville, A. N. (2006), The five greatest applications of markov chains, *in* 'Proceedings of the Markov Anniversary Meeting', Citeseer, pp. 155–158.

Wang, R. Y. & Strong, D. M. (1996), 'Beyond accuracy: What data quality means to data consumers', *Journal of management information systems* **12**(4), 5–33.

Yang, X., Xu, G., Li, Q., Guo, Y. & Zhang, M. (2017), 'Authorship attribution of source code by using back propagation neural network based on particle swarm optimization', *PloS one* **12**(11), e0187204.

Ye, J., Janardan, R. & Li, Q. (2005), Two-dimensional linear discriminant analysis, *in* 'Advances in neural information processing systems', pp. 1569–1576.

Appendix

Important Python Code (First order Markov chain)

```

character_allow = [ " ", "-", "'", '"', ".", ","]   ##Allowed Character "\n",
alphabet_allow = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y","z", " "]
def character_process(char):
    char = char.lower()
    if char.isnumeric():
        return " "
    if char.isalpha():
        if (char in alphabet_allow):
            return char
        else:
            return "  "
    if char in character_allow:
        return char
    else:
        return " "
```

Listing 5.1: Character selection

```
#Create transition matrix base on the path from csv file
class training_book:
    def __init__(self, book_path):
        self.book_path = book_path

    def create_transition_matrix(self):
        with open(self.book_path,  encoding="utf8", mode='r') as f:
            corpus=f.read()
        corpus=corpus.replace("\n", " ")
        char_list = []
        for char in corpus:
            clean_char = character_process(char)
            char_list.append(clean_char)

        str_char = ''.join(char_list)
```

```python
        clean_char=' '.join(str_char.split())

        #Sort the distinct letter
        distinct_words = character_allow + alphabet_allow  ## set of 33 characters
        distinct_words.sort()

        #create word2id and id2word dictionary
        word2id_dict = {}
        id2word_dict = {}
        for index, word in enumerate(distinct_words):
            word2id_dict[word] = index
            id2word_dict[index] = word

        #Initialize empty transition matrix
        M = N = len(distinct_words)  ## Limit to 33 characters
        transition_matrix = np.zeros((M, N))

        #Create 1st order Markov Chain (Number of occurance)
        for i in range(len(char_list)-1):
            current_word = char_list[i]
            current_word_id = word2id_dict[current_word]
            next_word = char_list[i+1]
            next_word_id = word2id_dict[next_word]
            transition_matrix[current_word_id][next_word_id] = transition_matrix[current_word_id][next_word_id] + 1

        sum_of_each_row_all  = np.sum(transition_matrix, 1)

        #Create 1st order Markov Chain (Number of occurance)
        for i in range (len(distinct_words)):

            single_row_sum = sum_of_each_row_all[i]

            if (sum_of_each_row_all [i] == 0):
                single_row_sum = 1

            transition_matrix[ i,: ] =  transition_matrix[ i,: ] / single_row_sum

        return word2id_dict, id2word_dict, transition_matrix
```

Listing 5.2: training book class that obtain the transition matrix

```python
## Calculate Log-likelihood
def Sum_log_likelihood(transition_matrix, test_path, word2id_dict):
    #log_likelihood = 0
    test_chapter=''
    with open(test_path,  encoding="utf8", mode='r') as f:
        test_chapter=f.read()
    test_chapter=test_chapter.replace("\n", " ")

    clean_test_chapter = []
    for char in test_chapter:
        clean_charr = character_process(char)
        clean_test_chapter.append(clean_charr)

    str_char = ''.join(clean_test_chapter)
    clean_test_chapter=' '.join(str_char.split())
    len_clean = len(clean_test_chapter)
    log_likelihood = np.zeros(0)
    for i in range(0, len(clean_test_chapter)-1):
        current_word = clean_test_chapter[i]
        next_word = clean_test_chapter[i+1]

        Step_probab = transition_matrix[word2id_dict[current_word] ,
    word2id_dict[next_word]]
        log_likelihood = np.append(log_likelihood, Step_probab)

    log_likelihood = np.log(log_likelihood)
    likelihood_neglect_special_case = 0

    inf_count = 0

    for i in range(len(log_likelihood)):
        if (log_likelihood[i]!= float("-inf")):
            likelihood_neglect_special_case = likelihood_neglect_special_case
    +log_likelihood[i]
        else:
            inf_count = inf_count+1

    if (inf_count!=0):
        print(test_path, inf_count, len(clean_test_chapter), len(
    clean_test_chapter)/inf_count)

    log_likelihood_acc = np.where(log_likelihood == float("-inf"), 0,
    log_likelihood)
    log_likelihood_acc = np.cumsum(log_likelihood_acc)
    return likelihood_neglect_special_case/(len_clean-inf_count),
    log_likelihood_acc
```

Listing 5.3: Log-likelihood model

```python
##KL & JS Divergence function
import math
import numpy

def kl_divergence(test_p, author_q, threshold = 0.2):
    result = []
    for i in range(len(test_p)):
        row_KL = []
        for j in range(len(test_p[i])):
            if (test_p[i][j] == 0):
                KL_div = 0
            elif(author_q[i][j] == 0):
                KL_div = threshold   ## Special case where q is 0
            else:
                KL_div = test_p[i][j] * math.log(test_p[i][j]/author_q[i][j])
            row_KL.append(KL_div)
        result.append(sum(row_KL))
    return result

def js_divergence(p, q, thres = 0.2):
    m = numpy.multiply(0.5, numpy.add(p, q))
    y = numpy.add(numpy.multiply(0.5, kl_divergence(p, m, thres)) ,numpy.
    multiply(kl_divergence(q, m, thres), 0.5))
    return y
```

Listing 5.4: KLD and JSD model

```

import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
clf = LDA()
clf.fit(X, y)
LDA(n_components=None, priors=None)
LDA_pred = clf.predict(test_TM)

book_aut = []
for i in range(len(book_path)):
    book_aut.append(book_path[i].split("/")[-1].split(".txt")[0])
```

Listing 5.5: LDA model

```
##1st order KL CM

y_true = True_aut
y_pred = KL_pred
data = confusion_matrix(y_true, y_pred)
df_cm = pd.DataFrame(data, columns=np.unique(y_true), index = np.unique(
    y_true))
df_cm.index.name = 'Actual'
df_cm.columns.name = 'Predicted'


plt.figure(num=None, figsize=(13, 13), dpi=150, facecolor='w', edgecolor='k')
sn.heatmap(df_cm, cmap="YlGnBu", annot=True, xticklabels=Simp_True_Label,
    yticklabels=Simp_True_Label)# font size
```

```
plt.savefig(r"C:/Users/Teng Li Yuan/Desktop/Dissertation result plot/1st
    order KL.png")

target_names = aut_orig
print(classification_report(y_true, y_pred, target_names=target_names))
```

Listing 5.6: Confusion matrix and class performance table