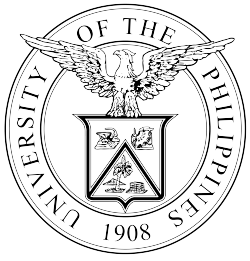
Text Analytics of the Qur'ān using Bayesian Statistics and Large Language Models

by

Al-Ahmadgaid Bahauddin Asaad

Submitted to the *Institute of Islamic Studies*
in partial fulfillment of the requirements for the degree of
Master of Arts in Islamic Studies



University of the Philippines Diliman
Diliman, Quezon City

May 2025

Contents

List of Figures

Abstract

The interest of the paper is to provide a comprehensive text analytics of the Qur'an. This is by utilizing Statistical and Machine Learning methods to computationally analyze the said scripture. Specifically, the following procedures have been done: descriptive analyses of the structure of the Qur'an, morphological analyses of the Qur'an. Further, the data used for the Qur'anic Arabic corpus is the one in QuranTree.jl by Asaad (2022). The computational software used is Julia programming language.

Chapter 1

Introduction

The use of scientific computing to studying the Qur'ān is still in its early stage in the fields of Islamic Studies and Statistical and Machine Learning applications. This is evident from the fact that there is no known books yet on this topic as far as the knowledge of the author at the time of writing. In addition to this, because this paper tackles methodology from both fields, it therefore benefit not only the researchers from Islamic Studies but also Statisticians and Machine Learning practitioners who are into text analytics. Having said that, it is necessary to provide context to audiences of these disciplines on both backgrounds on the state of Qur'ānic studies and the increasing adoption of scientific methodologies to studying humanities.

1.1 Background

The Qur'ān or *al-qurʾān* اَلقُرآن meaning *the recitation*, the holy book of Islam, is revered by 1.9 billion (according to 2020 projection of Cooperman et al. (2011, p. 13)) Muslims across the globe as the literal words of God. Muslims believed that the Qur'ān was gradually revealed (Qur'ān 25:32) to Prophet Muhammad ﷺ through angel *ǧibrīl* جِبرِيل or Gabriel (Qur'ān 2:97). The Qur'ān contains 77,429 Arabic words in total, which covers only 56 percent of the Greek New Testament which has 138,020 words in total (Sinai, 2017, p. 11).

The Qur'ān is divided into *sūwar* سُوَر (*plural* of *sūra* سُورَة) which are the equivalent of chapters, each containing *āyāt* آيَات (*plural* of *āya* آيَة meaning *signs*), which are the equivalent of verses. The *sūwar* سُوَر are not arranged in chronological order as in the Bible's books and chapters, but rather arranged in monotonically decreasing length of number of verses after the first *sūra* سُورَة (*see* Figure 1.1). The *sūwar* سُوَر of the Qur'ān can be categorized into two types: the *makkiyya* مَكِّيّة (Meccan) and *madaniyya* مَدَنِيّة (Medinan). The categories refer to the geographical lo-

cation of where the *sūra* سُورَة was revealed to Prophet Muhammad ﷺ. Figure 1.1 shows the groupings of the *sūwar* سُوَر. Note that some of the *sūwar* سُوَر have mixed geographical locations[1], that is, a few of the *āyāt* آيَات in it were revealed in other geographical location apart from the geographical location of the rest of the *āyāt* آيَات. Therefore, the categorization in Figure 1.1 highlights the geographical location of the majority of the *āyāt* آيَات in the *sūra* سُورَة.

Attempts at understanding the Qur'ān by Qur'ānic scholars were mostly done with the use of manual processes, that is, studying the scriptures by going through its content manually line-by-line or one-at-a-time. However, with the advent of computers, some researchers have started using it to aid in their study. The first

[1] *see* list of the location in `https://tanzil.net/docs/revelation_order`

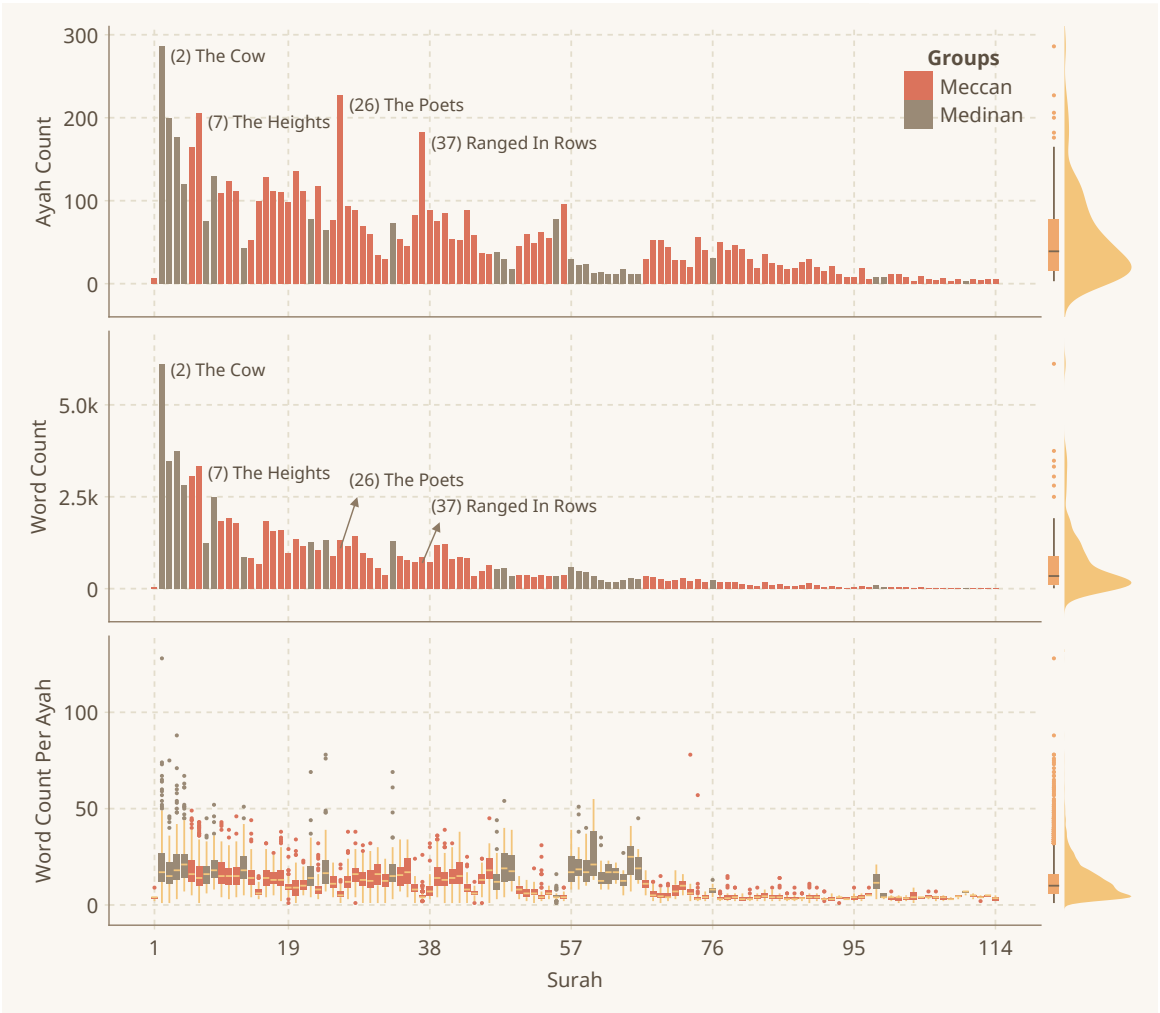


Figure 1.1: Statistics of the words and *āyāt* آيَات (verses) of the Qur'ān

known to have used computers for studying the Qur'ān was likely Rashad Khalifa in 1968[2], where he studied the significance of the mysterious initials at the beginning of some *sūwar* سُوَر. Rashad uploaded the Qur'ān into his computer by transliterating the Arabic letters and other Qur'ānic orthographies into Roman letters and symbols that the computer can easily parse. This approach of using computers to find new insights is more common in the field of science, and it was new to the field of Qur'ānic studies.

Indeed, to proceed with the use of scientific computing, the Qur'ān will be treated as the data that needs to be analyzed using a scientific process called Natural Language Processing (NLP), a branch of Machine Learning (ML) that aims to understand natural languages, such as Arabic. To instruct the computer to do Statistical analyses or ML, one needs to use a *software application* or a formal[3] language called *programming language*. There are several programming languages that the computer can understand. The popular one for researchers in the field of sciences are Python (Van Rossum & Drake Jr, 1995), R (R Core Team, 2023), and sometimes Julia (Bezanson et al., 2017). These programming languages will be used to construct instructions for computers. Therefore, if the data is the Qur'ān, then there should be a way to interface with it using any of these programming languages; or alternatively, there should be a way to upload it into the chosen programming language and encode the Arabic letters into something that can be easily parsed by the computers, like what Rashad Khalifa did. Having said that, there are indeed some programming languages with libraries or packages for interfacing with the Qur'ān, and this is true for Python, R, and Julia. For this study, the three programming languages will be used. The reason is that the scope and objective of this paper covers a wide range of statistical and Machine Learning methodologies, and that the three programming languages have advantages in certain methods and not on others. As such, the paper will use the appropriate programming language that

[2] `https://www.masjidtucson.org/quran/miracle/a_profound_miracle_sura68nun133.html`

[3] "formal" because these languages were invented by man for particular purpose, in this case to communicate with computers

is more fit for the task. The ruling is that Julia will be used for interfacing with the Qur'ān texts since its library for it has more features (*see* Asaad, 2021) compared to R and Python. The said Julia library is the QuranTree.jl[4]. QuranTree.jl is based on Tanzil[5] for the Qur'ānic Arabic texts, and Dukes and Habash (2010) for morphological annotation, which both libraries from R and Python do not have in terms of morphological annotations from Dukes and Habash (2010). On the other hand, both R and Python will be used for libraries that are not available in Julia. In particular, R is known for niche statistical libraries since it was made for statistical computation, whereas Python is now popular for Deep Learning frameworks for complex modeling like TensorFlow[6] (a library made by Google[7]) and PyTorch[8] (a library made by Meta[9]).

Given the programming languages, next is to understand how Statistics and Machine Learning can help in studying the Qur'ān. Statistics is a branch of Science that aims to study features or characteristics of data generated from a random phenomenon. The findings of Statistical analyses can then be used to make decisions, conclusions or predictions of the general population of the data or general characteristics of the data. Machine Learning or ML, on the other hand, is a branch of Artificial Intelligence that heavily intersects with Statistics, albeit with distinct differences as well. Both Statistics and ML aims to characterize data by learning its features, but ML researchers have been aiming on complex models that are often inspired by simpler models from Statistics. Therefore, one can think of Statistics as one of the fundamentals of ML.

One of the goals of Statistics and Machine Learning is to summarize all of the learned features of the data into a hypothesized equation or hypothesized mathematical formula by optimizing the parameters or weights of this equation or for-

[4] https://alstat.github.io/QuranTree.jl/stable/
[5] https://tanzil.net/download/
[6] https://www.tensorflow.org/
[7] https://research.google/
[8] https://pytorch.org/
[9] https://ai.meta.com/

mula to capture *most*[10] of the characteristics of the data. The idea stems from the fact that any data point is simply a sum of its components, which are factors known to affect the data point. For example, the volume of orders in a restaurant is dependent on the time of the day. More orders are expected during lunch and dinner. Hence, volume of orders as data points can be derived from the sum of factors such as time of the day, and also day of the week (assuming more orders during weekends or weekdays). Therefore, from this example, any data point is affected by factors that can be translated into math so that their sum results into the data points we want, like the volume of orders. It is for this reason why the hypothesized mathematical equations or formula are used for summarizing the features and capturing the characteristics of data. It is called *hypothesized equation* since the researcher is the one who decide which family of equations best describe the characteristics of the data. There are many possible mathematical formulae that can be constructed, and not all can be used to describe the data, this is why the researcher is the one who decide. So that, those equations that can be used for understanding the data are special enough that they are referred to as *models*. Mathematically and in general, a model can be related to the data as follows:

$$y = h(x|\boldsymbol{\theta}) + \varepsilon, \tag{1.1}$$

where $h(x|\boldsymbol{\theta})$ is the hypothesized model taking the input data x (from the examples discussed above, these are the values of the factors affecting in the volume of orders), where $x \in \mathcal{X}$ (from the examples discussed above, $\mathcal{X}$ is the set containing all values of the factors mentioned), and outputs a target data y (from the examples discussed above, this is the volume of orders), where $y \in \mathcal{Y}$. Therefore, $h : \mathcal{X} \rightarrow \mathcal{Y}$, meaning h maps the factors or features into the set of the target variable y. The ε is the error that the hypothesized model $h(x|\boldsymbol{\theta})$ cannot capture even after finding the best configuration of its parameter $\boldsymbol{\theta}$. Ideally the error or *residual* or sometimes

[10]Not all characteristics of the data can be captured by the model, and those that are not captured are called the errors, and the goal is to minimize the errors into something tolerable.

called *innovation*, ε, should exhibit a *random noise* for us to say that the hypothetical model $h(x|\boldsymbol{\theta})$ has captured well the core characteristics of the data (x, y). These random noise are data points that could have come from other factors that are not available in the given data, (x, y). Therefore, the idea of modeling is to find the optimal value of $\boldsymbol{\theta}$ such that the error between the actual data (represented by y below) and the predicted one $\hat{y}$, is as small as as possible or tolerable:

$$\varepsilon := y - \hat{y} = y - h(x|\boldsymbol{\theta}). \tag{1.2}$$

To help understand the concept of modeling, and relate it to the fashion industry, which the author assumes most readers are familiar with, a model in a fashion industry is responsible for representing the characteristics of the target customers (in Eq. 1.1 this is y). Therefore, for a clothing company, they hire Asian models (in Eq. 1.1, this is $h(x|\boldsymbol{\theta})$) to target Asian customers (in Eq. 1.1 this is y). So that, when these models wore the clothes sold by the said company, the potential customer will more or less be able to relate to the model, and be able to imagine themselves wearing that same clothes as well, which help them incline to buying the said clothing. The model, therefore, does not necessarily have the looks of every target Asian customers, but at least in terms of height, skin tone, hair, and other common Asian features, the model will likely have it, or at least the difference is more or less minimal (in Eq. 1.1, the difference is represented by $\varepsilon := y - \hat{y}$). The question now is, what are the benefits that this model can bring to the clothing company? Well, the clothing company will be able to create products that are tailored to their Asian customers using the said model, since the company will have the right baseline measurements needed. Relating this analogy to the technical concept discussed earlier, you can think of the target customers as the real or actual data (in Eq. 1.1 this is y), and the model as the same technical term use in Machine Learning and Statistics (in Eq. 1.1 this is $h(x|\boldsymbol{\theta})$), but this time this technical model is expected to capture the characteristics of the real data analogous to fashion model that is expected to capture the characteristics of the target customer. This Statistical or Machine Learn-

ing model brings the following benefits: researchers will be able to study the real data by simply using the model to answer questions that are not available in the sampled real data.

1.2 Rationale of the Study

Having understood the background of this study, let's turn our attention to the rationale. When it comes to modeling the characteristics of any data, there are several ways to do it, and it depends on the richness of the data and the questions one would like to answer. This is true for the Qur'ān. The rich morphological annotations done by Dukes and Habash (2010) has opened a lot of opportunities for statistical analyses. It is even hard on where to start since this morphological data have not been extensively studied yet, at least based from the survey of Bashir et al. (2023) and Darwish et al. (2021). Most of the natural language processing studies done on the Qur'ān have revolved around thematic clustering of some *sūwar* سُوَر (Siddiqui et al., 2013); semantic techologies for improved keyword search on the Qur'ān (Afzal & Mukhtar, 2019); Qur'ān recitation through speech recognition (Abro et al., 2012; Ahmed & Abdo, 2017); Qur'ān grammatical analyses by predicting the parts-of-speech (POS) (ELAffendi et al., 2021), *iᶜrāb* إِعرَاب modeling through enhanced context-free grammar (Mannaa et al., 2022); Semantic systems for the Qur'ān, such as the use of doc2vec based vectorization of the *āyat* آيَت (Alshammeri et al., 2021).

The aim of this paper is to make use of the Statistical or Machine Learning modeling to understand the characteristics of the Qur'ān. More specifically, the next section will list down the research questions of this paper.

1.3 Objectives

The following are the general and specific objectives of this paper:

1. What are the structural characteristics of the Qur'an that can be extracted from its rich morphologies using statistical and large language models?

    (a) What are the statistics of the Qur'ān's morphological features in terms of its parts of speech and selected entities like God's name and the prophets

names mentioned?

(b) How do the rhythmic signatures of the Qur'ān of the verses looks like and what are statistical insights that can be extracted?

2. What other insights that can be extracted from the semantics of the Qur'an's texts using statistical and large language models?

   (a) How does the theory of *concentrism* be formulated statistically, and what are the insights from the statistical and large language models on this?

   (b) How do the surahs are organized in terms of the topics? What are the themes that can be extracted for each of the surahs?

   (c) How do these extracted themes compare to the summaries of Abdel Haleem's English translation of the Qur'an?

3. How does this combination of statistical, machine learning, and artificial intelligence with the Muslim's traditional literatures help in understanding the Qur'ān, especially with the advent of Generative AI?

1.4 Significance of the Study

While the Qur'ān has been extensively studied by Muslims and non-Muslims scholars alike, especially in the topic of Meccan and Medinan surahs, there is still a lot to uncover from the perspective of Computational Statistics. Hence, the significance of this study is that it brings forward new ways of extracting insights from the Qur'ān by leveraging Computations, Statistics, Machine Learning, and AI, that is still in its early stage in the field of Qur'ānic Studies. Therefore, this new perspective or process of studying the scripture not only aids the scholars of the Islamic Studies, but may also contribute indirectly to community development and policy makers who use Qur'ān as part of their decision making.

1.5 Scope of the Study

The paper will cover all chapters of the Qur'ān both for the Morphological and Topic Modeling analyses, but it will only present the results of *sūrahs* سُور with at

least 1000 words. The rest of the result will be part of the web application that can be used to query the Qur'ān. It will also not delve into the *tafsir* تَفسِير of each of the verses, but only when necessary for further context.

1.6 Thesis Organization

The paper is organized as follows: Chapter 2 will discuss the related literatures, Chapter 3 will discuss the methodology, Chapter 4 will present the results and discussions, and finally Chapter 5 will contain the conclusion and recommendation. The references and appendices are placed after the Chapter 5.

Chapter 2

Literature Review

As mentioned in the previous chapter, the earliest work in Qur'ānic studies using computer was likely the work of Rashad Khalifa in 1968[1], which led to one of his book entitled 'The Computer Speaks: God's Message to the World' (*see* Khalifa (1981)). While the work of Rashad started at studying the mystery letters in the beginning of some *sūrahs* سُور (for example Qur'ān 2:1, 3:1, 7:1, etc.), it quickly went on to cover what he calls other *mathematical miracles*, all of which are covered in Khalifa (1981). His findings led him to generalize the claim that God revealed His words through this mathematical patterns throughout the Qur'ān, and that those verses that were off and did not conform to this discovered mathematical patterns led him to extensive investigation of the said verses, and concluded that those could be or surely be an insertion that should not have been in the Qur'ān in the first place. There are two verses that were off according to Rashad, and he called these verses as *false verses*[2], these are the last two *ayāt* أَيَات of *sūra l-tāwba* سُورَة التَّاوبَة or The Chapter of *Repentance*. These two verses were removed in Rashad's Qur'ān translation[3]. Rashad believed so much on his findings that he self-proclaimed himself to be a messenger[4] with this new findings and that the Qur'ān nowadays should conform to his found mathematical patterns. This self-proclamation led to his assassination.

Fast forward to 20th century, among the pioneers to creating a stemmer systen for the Qur'ān is the work of Thabet (2004). A stemmer system is a system for trimming inflected words into its basic form, which grammatically mean its the root form. For example, in English language the root word for *computational*, *computer*, *computation*, and *computerize* is *compute*. Therefore, from the root of the word forms different stems representing the different words. Hence, the idea of stemming is to

[1] https://www.masjidtucson.org/quran/miracle/a_profound_miracle_sura68nun133.html
[2] https://submission.org/App24.html
[3] https://www.masjidtucson.org/quran/frames/
[4] https://www.masjidtucson.org/submission/faq/rashad_khalifa_summary.html

trim these words into its basic form, so that it would be easy to do word clustering or grouping through word similarity. According to Thabet (2004), the rich morphology of the Qur'ānic language or the Classical Arabic makes it even more difficult to do word stemming. Moving on, Thabet (2005) builds on this stemming system, and used it for tokenization of the Qur'ānic words. Tokenization is the process of listing all of the words in a sentence, and Thabet (2005) makes use of Thabet (2004) to further cleanse the noise from these tokens brought by the morphological variants of the Classical Arabic words. After cleansing the data, Thabet (2005) makes use of a statistical methodology for clustering or grouping the chapters of the Qur'ān, in particular the statistical methodology used is the Agglomerative Hierarchical Clustering based on the Euclidean distance of the adjusted word frequency of the *sūrah* سُورَة.

The work of Thabet (2004) and Thabet (2005) makes use of the Qur'ān corpus transliterated to Roman letters and symbols. Indeed, with interest growing on studying the Qur'ān from the lense of Data Analysis and Natural Language Processing, more work have been put in place into creating digital corpi of the Qur'ān that captures the different aspects of its linguistic styles. Hence, a series of work by Sharaf and Atwell led to the following publications: Sharaf and Atwell (2009) studied knowledge representation of the Qur'ānic verb valences using FrameNet frames, the output of which is a lexical database of the corpus of Qur'ān verbs. Further, the work of Sharaf and Atwell (2012a) came up with corpus for the annotations of the Qur'ānic pronouns, the authors named it as QurAna. Building on this work, Sharaf and Atwell (2012b) came up with a corpus for studying Qur'ānic relatedness based on the commentary of Ibn Kathir اِبن كَثِر, the authors named this corpus as QurSim.

Moving on, an unpublished work by Nassourou (2011) used a Support Vector Machine (SVM) to predict the classification of the place of revelation of the Qur'ānic *sūrahs* سُور. The idea was to use *sūrahs* سُور with well attested place of revalation as the training set, and then train an SVM to predict the remainings of the chapters.

Furthermore, the work of Shahzadi, ur Rahman, and Sawar (2012) developed a simple classifier based on the frequency distribution of words in *ayāt* أَيَات and the corresponding words in *sūrahs* سُور.

Moving on, the work of

Chapter 3

Historical Background of the Qur'ān

The Qur'ān (meaning *the recitation*) was revealed *orally* by angel *ǧibrīl* جِبرِيل to Prophet Muhammad ﷺ and passed onto other believers through oral tradition (reciting the Qur'ān to students repeatedly so as to memorize it, instead of writing it down and let the believers read it and memorize it). Memorizing 77,429 Arabic words of the Qur'ān through oral transmission can be a difficult task, but what aids this memorization is the rhythm feature of the Qur'ān. According to one Orientalist, Sinai (2017), "rhyme, however, or rather a periodically recurrent word-final assonance, is a feature of the Qur'ān throughout, and it naturally partitions the *sūrah* سُورَة." Indeed, because of this feature, it makes it easy to memorize the entire Qur'ān, and the one who do so is called *hafiẓ* هَفِظ meaning *one who remembers* or *keeper*. Qur'ān memorization contest is a common event in Muslim countries, the Philippine embassy has hosted one in 2022 in Saudi Arabia (Manila Bulletin, 2022).

According to the Muslim tradition, the oral transmission of passing the Qur'ān from a *hafiẓ* هَفِظ to new believers was gradually put into writings as requested by the believers themselves. The idea was brought up after the battle of Yamama, where many of the Muslims who died were *qurrāʾ* قُرَّاء (*the one who properly recite the Qur'ān*), and so fearing that their numbers will reduce in other battle fields, Umar ibn al-Khattab عُمَرُ بنُ الخَطّاب (who became the second caliph) suggested to the first caliph, Abū Bakr 'Abd Allāh ibn 'Abī ibn 'Abī Quḥāfa, أَبُو بَكرٍ عَبدُ اللّاهِ إِبنِ أَبِي قُحَافَةَ or short for Abū Bakr أَبُو بَكر, to collect the Qur'ān into writing. Abū Bakr then authorized Zaid ibn Thabit زَيدِ بنِ ثَابِتٍ for the task. According to Zaid, he started collecting from the leafless stalks of the date-palm tree and from the pieces of leather and hides and from the stones, and from the chests of men (who had memorized the Qur'ān, i.e. the *hafiẓ* هَفِظ)[1]. Long story short, the effort was finally codified by the third caliph, Uthman ibn Affan عُثمَانُ بنُ عَفّان in the year 645 CE, which was then recopied and

[1] *see* `https://sunnah.com/bukhari:7191`

distributed to the different regional capitals of the early Islamic empire of that time. The rest of the copies outside this codification were then burned[2] down in order to have one standard Qur'ān. The Qur'ān nowadays is therefore assumed to be based on Uthmanic codex because of the story mentioned. That is, if indeed Uthman has ordered to burn other copies of the Qur'ān outside his codification, then what's left should only be based on his codex or archetype, and that should only be the inherited codex of the Muslims today.

The Qur'ān is believed by the Muslims to have been preserved since it was first recited by angel *ǧibrīl* جِبرِيل or Gabriel to the Prophet ﷺ. Many orientalists had been skeptic about this claim, for example, John Wansbrough theorized that the Qur'ān was collected over a 200-year period (*see* Wansbrough, 2004, p. 101) after the death of the Prophet ﷺ, instead of within a few years after the death of the Prophet ﷺ. However, recent findings through radiocarbon dating brings forward strong evidence of potential preservation of the whole Qur'ān, which the Muslims believed to be so. For example, the Birmingham Qur'ān manuscript discovered in 2015 is dated to be between 568 and 645 CE with 94.5% accuracy, making it among the oldest Qur'ānic manuscript in the world (*see* Birmingham University, 2015). Its predicted range of years intersects with the lifetime (570 to 632 CE) of the Prophet ﷺ. What is interesting is that the Birmingham Qur'ān is consistent with the Qur'ān today, word-by-word and letter-by-letter[3], see Figure 3.1. This is indeed another evidence that the Qur'ān today was codified by Uthman since the discovery of the Birmingham Qur'ān manuscripts have confirm it. In addition to this, the Sana'a Palimphest is also among the oldest Qur'ān radiocarbon dated to be between 578 CE and 669 CE with 95% accuracy (Sadeghi & Bergmann, 2010), which according to Sinai (2017), "neither does the edited portion of the Sana'a palimpsest offer evidence for additional or missing verses or for a divergent verse order within the

[2] *see* `https://sunnah.com/bukhari:4987`

[3] The orthography of the Arabic letters in the early days had no diacritics and were written in its basic consonantal skeleton. That is, Arabic orthography and grammar were in their nascent when the Qur'ān was revealed, and therefore has to adjust and catch up, to capture and preserve the proper recitation of the Qur'an.

sūrahs سُور." Given these discoveries on the recent Quranic manuscripts, the claim of Wansbrough (2004, p. 101) is now untenable (*see* Sinai, 2014).

One likely reason as to why the scribes were able to preserve the Qur'ān in the two folios of the Birmingham Qur'ān manuscripts follows from the fact that the Qur'ān is firstly memorized before it was decided to be written. So that, the Topkapi manuscripts that contain 99% (only 23 verses missing)—dated around 701 to 750 CE (Karatay, 1962)[4], that is, around 69 to 118 years after the death of the Prophet ﷺ—of the Qur'ān today was likely partly written from memory. This is possible since the Qur'ān possesses a rhytmic feature (that aids with memorization) that naturally divides its verses or *ayāt* أَيَات, and that the Qur'ān memorization competition is still held to this day (as in the example of Manila Bulletin, 2022), apart from

[4]*see also* https://corpuscoranicum.de/en/manuscripts/1977/page/1-410?sura=1&verse=1

Figure 3.1: 20th Century Qur'ān (left) in its fully featured orthographies vs Birmingham Qur'ān dated between 568 and 645 CE (right) in its basic consonantal skeleton. Image from Wikipedia (2015).

the fact that it needs to be recited (any chapter after the first chapter of the Qur'ān according to the choice of the worshipper) every prayer from memory. Bottomline, there are many avenues for Qur'ān recitation from memory, and these have helped in its preservation. Moreover, since there are no significant evidence of insertion or malicious intention on addition or revision in all of the extant Qur'ānic manuscripts so far, some Orientalists came up with other theories of insertions on the basis of the literary style of the Qur'ān, see for example Sinai (2017, p. 92), where verse 102 of *sūra l-ṣāffāt* سُورَة الصّافّات or The Chapter of *Ranged in Rows* is theorized as addition because it is longer compared to other verses in the said chapter, refer to Sinai (2017, p. 92) for his other reasonings. Nonetheless, the Qur'ān is indeed stable based on the extant manuscripts.

Furthermore, the vastness of the early Islamic empire meant that different Muslim regional capitals have covered populations with different Arabic dialects, and so to accommodate these differences, Muslims believed that there were seven variant readings of the Uthmanic codex. Variant readings are defined as different pronunciations of the same word, in this case seven Uthamnic Qur'ān for seven different pronunciations. The *ḥadīṯ* حَدِيث or *narration* comes from Ubayy ibn Ka'b أُبَيّ بِن كَعب who reported[5] that the Prophet ﷺ was near the tank of Banu Ghifar that *ǧibrīl* جِبرِيل or Gabriel came to him and said: "... Allah has commanded you to recite the Qur'ān to your people in *seven dialects*, and in whichever dialect they would recite, they would be right." Recent work of Sidky (2020) shows that the material evidence on the regional variants is in remarkable agreement with well-attested written variants documented in the traditional Muslim literature.

Muslim and non-Muslim scholars alike have been extremely interested in understanding the unique literary characteristics of the Qur'ān. As mentioned earlier, unlike other books like the Bible (arranged in chronological order), the Qur'ān does not follow any obvious organization. In addition to this, a *sūrah* سُورَة does not fit the exact definition of a chapter. Indeed, the name attached to a *sūrah* سُورَة is often de-

[5]source: `https://sunnah.com/muslim:821a`

cided as the unique entity mentioned in the said *sūrah* سُورَة, its main purpose is to help early Muslims distinguish which *sūrah* سُورَة they are talking about, this is contrary to the chapter name where the associated name is obviously the main topic of the chapter. Further, as described by Sinai (2017), "... the compositional unity of the long surahs located at the beginning of the corpus is anything but obvious: at least at first sight, they can appear a flit back and forth between different topics in a largely haphazard manner. This impression is not limited to Western readers: even pre-modern Muslim scholars have often approached their scripture as a quarry of unconnected verses and groups of verses that bear little intrinsic relation to what precedes and follows." It wasn't until Neuwirth (2007), that the compositional unity of the Qur'ān can be observed in tighter literary unities, as Neuwirth (2007) showed that the many of these texts display a tripartite structure and are often constructive around a narrative middle part (Sinai, 2017). Samples of the organizational style of the Qur'ān was shown in Sinai (2017, p. 88).

Chapter 4

Background on Probability and Statistics

This chapter will discuss some statistical concepts that will be used to understand and build up the ideas behind the methodology of this paper, which is presented in the next chapter. With that said, the discussions are organized as follows: Section 4.1 will present the concept of Descriptive Statistics; Section 4.2 will discuss the Probability Theory; and, Section 4.3 will discuss the Statistical graphs or plots.

Further, the topics discussed here may be self-explanatory for Statisticians, ML researchers or those with Mathematical background. However, for the benefit of readers coming from humanities background, the paper will present the methodology as follows: mathematical formulas are formalized for purpose of brevity through Definition, Proposition, and Corollary, but immediate to these are explanations or examples aimed to be simple enough for non-statistician readers. As a guide, statisticians or ML researchers can simply read the Definition, Proposition, and/or Corollary. Whereas for humanities readers, the reading shouldn't not stress too much on the said mathematical formalities, and instead proceed to the discussions or examples immediate to it to aid with the understanding.

4.1 Descriptive Statistics

Among the basic statistical methodologies for summarizing information or data is what is known as Descriptive Statistics. From the name itself, these statistics are meant to convey simple descriptions of the data. For example, *mean* and *variance* are common statistics used for describing the data. The formulas for these statistics are given in the following definitions:

Definition 4.1.1 (Mean)**.** Let $x_i, i \in \{1, \cdots, n\}$ where $n \in \mathbb{N}$, then the *mean* of x_is is defined as follows:

$$\bar{x} = \sum_{i=1}^{n} x_i, \qquad \text{where } x_i \in \mathbb{R}. \tag{4.1}$$

⊸

Definition 4.1.2 (Variance)**.** Let $x_i, i \in \{1, \cdots, n\}$ where $n \in \mathbb{N}$ and let $\bar{x}$ be the mean defined in Defn. 4.1.1 then the *variance* of x_is is defined as follows:

$$\mathbb{V}\text{ar}(x) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2, \qquad \text{where } x_i \in \mathbb{R}. \tag{4.2}$$

⊸

Definition 4.1.3 (Standard Deviation)**.** Let σ^2 be the variance, then the standard deviation, denoted by σ, is defined as $\sigma := \sqrt{\sigma^2}$, that is, square root of the variance.

⊸

The mean is simply the average of the data points, while the variance is a single number that measures or summarizes the distances of the data points from the mean. The variance therefore measures how spread or varied the data points are. In practice, however, standard deviation is more popular for measure of variability since it doesn't get big easily due to the square root operator. Standard deviation is also the preferred metric for this paper.

4.2 Probability Theory

Statistics is built around the concept of Probability Theory. Hence, it is important to understand how this mathematical theory behind the statistical concepts work. Probability theory is a discipline in itself. It is a branch of mathematics that aims at studying realizations or observations from random phenomena. It does this by using algorithms and models (*see* discussion in Chapter 1 for understanding the model). These probability models are mathematical formula design to characterize or describe the patterns of data points. The simplest form of these models is the univariate probability density functions. However, before discussing these models, the idea behind it should be build up from ground up so that humanities readers can appreciate it. To do this, the discussion will proceed with the concept of probability.

Definition 4.2.1 (Probability)**.** A probability is a mathematical concept that measures the likelihood or chance of some event to happen. ⊸

Indeed, this idea of probability is well known or is easily understood by many. So that, the probability that someone will die is 1, meaning 100%, since that's how natures and biology work. Further, the probability of selecting one *ayāt* أَيَات from *sūra l-fātiḥati* سُورَة الفَاتِحَت through a random draw is one out of seven, assuming each *ayāt* أَيَات has equal chances of being drawn.

Now, to gradually formalize the concept into mathematics, the succeeding definitions will be discussed to build up the definition of a probability distribution.

Definition 4.2.2 (Sample Space)**.** Let $\omega_1, \cdots, \omega_n, \forall n \in \mathbb{N}$ be the list of all possible outcomes of a random event or a phenomena, then the sample space, denoted by Ω, is defined as the collection of all these possible outcomes including the empty set denoted by $\emptyset$. ⊸

Example 4.2.1. Consider the random phenomenon or event where someone randomly picks a verse from the Qur'ān, what is the probability that it will be a Meccan مَكِّيَّة or a Medinan مَدَنِيَّة verse?

The possible output for such random event is either Meccan or Medinan. Suppose, we let Meccan to be denoted by ω_1 and Medinan to be denoted ω_2, then the sample space, which is the collection of all possible output is denoted as follows:

$$\Omega := \{\text{مَدَنِيَّة}, \text{مَكِّيَّة}\} = \{\text{Meccan}, \text{Medinan}\} = \{\omega_1, \omega_2\} \tag{4.3}$$

It should be understood that $\emptyset$ is also included in Ω, but is not written for brevity.

⊸•

Definition 4.2.3 (Event)**.** Let $\Omega = \{\omega_1, \cdots, \omega_n\}, \forall n \in \mathbb{N}$, be the sample space, then an event, denoted here as $\mathscr{A}$, is defined as the subset of the sample space, i.e., $\mathscr{A} \subseteq \Omega$. ⊸

Example 4.2.2. From Ex. 4.2.1, consider drawing two samples from the Qur'an, what is the sample space and give an example of a possible event?

Solution: The sample space is given below:

$$\begin{aligned}\Omega =&\{(\text{مَكِّيَّة}, \text{مَكِّيَّة}), (\text{مَكِّيَّة}, \text{مَدَنِيَّة}), (\text{مَدَنِيَّة}, \text{مَكِّيَّة}), (\text{مَدَنِيَّة}, \text{مَدَنِيَّة})\}\\ =&\{(\text{Meccan}, \text{Meccan}), (\text{Meccan}, \text{Medinan}), \qquad (4.4)\\ &(\text{Medinan}, \text{Meccan}), (\text{Medinan}, \text{Medinan})\} \qquad (4.5)\\ =&\{(\omega_1, \omega_1), (\omega_1, \omega_2), (\omega_2, \omega_1), (\omega_2, \omega_2)\} \qquad (4.6)\end{aligned}$$

So that, if $\mathscr{A}$ is the event of drawing two *ayāt* أَيَات from the Qur'ān, then a possible event is given by

$$\mathscr{A} := \{\text{مَكِّيَّة}, \text{مَدَنِيَّة}\} = \{\text{Medinan}, \text{Meccan}\} = \{\omega_2, \omega_1\} \qquad (4.7)$$

Therefore, $\mathscr{A} \subseteq \Omega$, read as $\mathscr{A}$ is a subset of Ω. ⊸•

Now, going back to the discussion on the concept of probability above and reflect on the example given, that the probability that someone will die is 1; and that the probability of selecting one *ayāt* أَيَات from *sūra l-fātiḥati* سُورَة الفَاتِحَت through a random draw is one out of seven. It therefore begs the following questions: how does one solve this? Like how does it translate into a formal mathematical computation?

Indeed, to appreciate the motivation of the succeeding definitions, it is important to devise a logical approach to computing probability mathematically, and this should explain why the following definitions are defined the way they are.

Probability as already defined in Defn. 4.2.1 is a measure, which will be formalized in Defn. 4.2.5. Indeed, this is analogues to measuring an object's size using a tape measure. So, when someone attempts to measure an object's size, there are conditions for the space of the object to be measurable. The first expectation is that, the space or area can be measured in several ways. Either measuring it as a whole, or measuring it piece by piece from its partitions. Now, measuring it as a whole using tape measure should be straightforward. However, measuring it by pieces can have several cases, and all of these should end up to the same measuring. These

cases accounts for the fact that when dealing with pieces of the surface one can start with different sizes of the pieces to be measured. So that, all the collections of all these possible configurations of pieces are mathematically called σ-algebra, and this collection should include the following:

1. The 0 size piece, that is, the object's measurement should start at 0;
2. The remainings of the pieces given the pieces already measured;
3. The union of all the pieces.

The above explanation for the σ-algebra is condensed in to the following mathematical definition:

Definition 4.2.4 (σ-algebra)**.** Let $\Omega := \{\omega_1, \cdots, \omega_n\}, \forall n \in \mathbb{N}$, then the collections of all disjoint partitions, which are the events, are defined as the σ-algebra or σ-field, denoted here as $\mathfrak{F}$, and should satisfy the following conditions:

1. The empty set $\emptyset \in \mathfrak{F}$
2. If $\mathscr{A} \in \mathfrak{F}$, then the complement $\Omega \backslash \mathscr{A}$ is also an element of $\mathfrak{F}$
3. If $\mathscr{A}_1, \mathscr{A}_2, \cdots$ is a countable sequence of sets in $\mathscr{A}$, then the $\bigcup_{i=1}^{\infty} \mathscr{A}_i$ is also an element of $\mathfrak{F}$

⊸

Example 4.2.3. Given the sample space $\Omega := \{\text{Meccan}, \text{Medinan}\}$, the σ-algebra is

$$\begin{aligned}\mathfrak{F} = \{&\text{Meccan}, \text{Medinan}, (\text{Meccan}, \text{Medinan}),\\ &(\text{Meccan}, \text{Meccan}), (\text{Medinan}, \text{Meccan}),\\ &(\text{Medinan}, \text{Medinan}), \emptyset\}\end{aligned} \tag{4.8}$$

⊸

Definition 4.2.5 (Probability Measure)**.** Let Ω be the *sample space*, $\mathfrak{F}$ be the σ-algebra, p be the probability measure, then the probability of a set $\mathscr{A}$, $\mathscr{A} \in \mathfrak{F}$, on the probability space $(\Omega, \mathfrak{F}, p)$, denoted by $p(\mathscr{A})$, satisfies the following properties:

1. Non-negativity: For any set $\mathscr{A}$, $p(\mathscr{A}) \geq 0$
2. Normalization: $p(\Omega) = 1$
3. Countable additivity: For any sequence of disjoint sets $\mathscr{A}_1, \mathscr{A}_2, \cdots$ in $\mathfrak{F}$, such that the union $\bigcup_{\forall i \in \mathbb{N}} \mathscr{A}_i$ is also in $\mathfrak{F}$, we have: $p\left(\bigcup_{\forall i \in \mathbb{N}} \mathscr{A}_i\right) = \sum_{\forall \in \mathbb{N}} p(\mathscr{A}_i)$

$\multimap$

Basically, the idea of Defn. 4.2.5 is to define the concept of "measure" in general sense, although above is a special measure called probability measure. As before, one can think of this probability measure like a tape measure in simple sense. This tape measure has some properties that should be expected for a measurement tool. The first property or condition is that the probability measure has to be positive. Indeed, if we use any tape measure for measuring length, never will someone get a negative measure like -2cm length. It always has to be positive. Further, the second condition to expect is that for this type of tape measure called probability measure, the total measure of all objects available in the given space should be equal to 1. That is, the total is normalized to 1. Think of this like 100% coverage if we measure all of the objects. Lastly, for any object, this tape measure should be able to measure the object through partitions, such that the measure of the union of these partitions is equal to the sum of the measure of each partition. All of these conditions are logical criteria for a general idea of "measure," although the normalization part above is unique for probability measure. Note that, the concept of proability measure here should not be confined to measuring length as in the analogy, it should generalize to measuring volume and complex objects in general.

Definition 4.2.6 (Random Variable)**.** Let Ω be the sample space, and $\mathbb{R}$ be the set of all real numbers, then a random variable X is a function defined as $X : \Omega \to \mathbb{R}$. $\multimap$

Example 4.2.4. Consider the example of drawing a random verse or *ayāt* أَيَات again, what is the probability that it will be an *ayāt* أَيَات from *sūrah l-baqara*'s *ayāt* آيَةُ سُورَة البَقَرَة or the Chapter of Cow?

The answer to this is 4.59% probability, this is because there are 286 *ayāt* أَيَات and there are 6236 verses in the Qur'ān, so that the probability is $\frac{286}{6236} = 0.04586$. The assumption here is that all of the *ayāt* أَيَات in the Qur'ān have equal chances of being picked up or drawn.

⊸

Example 4.2.5. To apply the concept so far, consider again Ex. 4.2.4, what is the probability of getting 5 *sūrah l-baqara*'s *ayāt* آيَةُ سُورَة البَقَرَة if we randomly pick 20 *ayāt* أَيَات in total from the Qur'ān?

Solution. The following are given:

- $n = 20$ independent trials of drawing $x = 5$ أَيَات from the Qur'ān
- Each trial has two possible outcomes: *naᶜam* نَعَم meaning Yes or *lā* لَا meaning No. That is, if the *ayāt* أَيَات is from the *sūrah l-baqara* سُورَة البَقَرَة then its نَعَم, otherwise لَا.

Now, the sample space consists of all possible sequences of 20 نَعَم and لَا. That is,

$$\Omega = \{(\text{لَا}, \text{نَعَم}, \text{لَا}, \cdots, \text{نَعَم}), \tag{4.9}$$

$$(\text{لَا}, \text{نَعَم}, \text{لَا}, \cdots, \text{لَا}), \tag{4.10}$$

$$\vdots \tag{4.11}$$

$$(\text{لَا}, \text{لَا}, \text{لَا}, \cdots, \text{لَا})\}. \tag{4.12}$$

In total, there are $20^2 = 400$ possible samples in the sample space Ω. Further, from Ex. 4.2.4, the probability of getting a *sūrah l-baqara*'s *ayāt* آيَةُ سُورَة البَقَرَة is 0.0459 or 4.59%. Therefore, if X is the random variable of an event of drawing an *ayāt* from the Qur'ān, if we assign لَا and نَعَم as either 0 or 1, respectively, then this would imply

that mathematically $p(X = 1) = 0.0459$. In addition, the probability of getting an *ayāt* أَيَات from other *sūrah* سُورَة would be

$$p(X = 0) = 1 - p(X = 1) = 1 - 0.0459 = 0.9541. \tag{4.13}$$

Further, let Z be the random variable for the event of getting n نَعَم from 20 trials, then the problem is now equivalent to finding the number of ways to choose n possible positions out of 20 in the collection or set. This can be solved using the *combination* formula as shown below:

$$\binom{n}{x} = \frac{n!}{r!(n-r)!} \Rightarrow \binom{20}{5} = \frac{20!}{5!(20-5)!} = 15,504. \tag{4.14}$$

That is, there are 15,504 possible cases of 5 positions' configuration for نَعَم in a 20 trial. Moreover, in each of these samples 5 has a probability of $p(X = 1) = 0.0459$, while the remaining 15 has a probability of $p(X = 0) = 0.9541$. So that,

$$\begin{aligned} p(Z = 5) &= 184,756 \times p(X = 1)^5 \times p(X = 0)^{20-5} \\ &= 15,504 \times 0.0459^5 \times 0.9541^{20-5} \\ &= 0.0016. \end{aligned} \tag{4.15}$$

Hence, there is a 0.16% probability of getting 5 *sūrah l-baqara*'s *ayāt* آيَةُ سُورَة البَقَرَة when randomly drawing 20 أَيَات from the Qur'ān.

⊸

Note that Ex. 4.2.5 can be solved using a known formula in probability called *Binomial* mass function, which is a model that can be used to describe the event of getting 5 *sūrah l-baqara*'s *ayāt* آيَةُ سُورَة البَقَرَة on a 20 random samples of Qur'ān's أَيَات. The Binomial mass function is specifically a probability mass/density function model. The following section will discuss the Statistical Graphs, which will cover the concept of frequency distribution, the one modeled or characterized by the probability mass/density function.

4.3 Statistical Charts

Charts or plots or graphs are data visualization tools that are useful for exploratory data analysis apart from the Descriptive Statistics discussed above. It supplements the Descriptive Statistics findings through shapes visualized in the graphs. Among the popular statistical graphs is the bar graph. An example of this is given in Figure 1.1. Other graphs used are the box plots and the density plots which is also in Figure 1.1

4.3.1 Box, Density, and Histogram Plots

While most statistical plots are easy to understand like bar graphs and scatter plots, others like Box, Density, and Histogram plots may not be easy to comprehend for someone with no Statistical background. This section will discuss how it is interpreted. Let's use Figure 1.1, Figure 4.1 for easy reference in this section.

As shown in Figure 4.1, both the Box and Density plots are tied to each other. This is indeed the case because both are describing the same information but presented in different style of visualization. In fact, Histogram is also used to describe the same information as the Box and Density, and the three are therefore related. So much so, that the three can be put into one graph. As to how to interpret these, readers are referred to for further details [to add reference].

Example 4.3.1 (Frequency Distribution)**.** Consider again the task of drawing an *ayāt* أَيَات from the Qur'ān, suppose the *ayāt* أَيَات are separated into مَكِّيَّة Meccan and مَدَنِيَّة Medinan, what is the probability of getting at most 10 *kalimāt* كَلِمَات or words in a sampled *ayāt* أَيَات from مَكِّيَّة Meccan *surah*s سُور?

Solution: To answer this, Figure 4.2 shows the *histograms* with the *box plots* and the *rainclouds* plots. The figure shows the frequency of the *kalimāt* كَلِمَات or words in a sampled *ayāt* أَيَات. This frequency describes the distribution of the *kalimāt* كَلِمَات. To interpret this, the Medinan مَدَنِيَّة histogram shows that most of the *ayāt* أَيَات have about 10 to 20 *kalimāt* كَلِمَات or words in total. This conclusion is based on the where the box of the box plot is located, which also corresponds to the area where the bars

of the histogram are high, and also where most of the points or 'droplets' from the rainclouds plot are congested. With that said, *historgram*, *box plot*, and *rainclouds* are related and are telling the same story from different perspectives. It should be noted that, the rainclouds plot is not a common visualization tool.

Now, comparing the numbers from Medinan مَدَنِيَّة to the *āyāt l-makkiyyatu* آيَات المَكِّيَّةُ, there are about 5 to 15 *kalimāt* كَلِمَات to expect per *ayāt* أَيَات based on Figure 4.2.

The question has not been answered yet though, the above discussions only explains how to interpret the graphs in Figure 4.2. So to answer the question, one simply needs to total the number of *āyāt* آيَات with at most 10 *kalimāt* كَلِمَات or words and divide this with the total number of *āyāt l-makkiyyatu* آيَات المَكِّيَّةُ. The answer is as follows, and this is part of the result of this paper: there are 4613 *āyāt l-makkiyyatu* آيَات

Figure 4.1: Statistics of the words and *ayāt* أَيَات (verses) of the Qur'ān

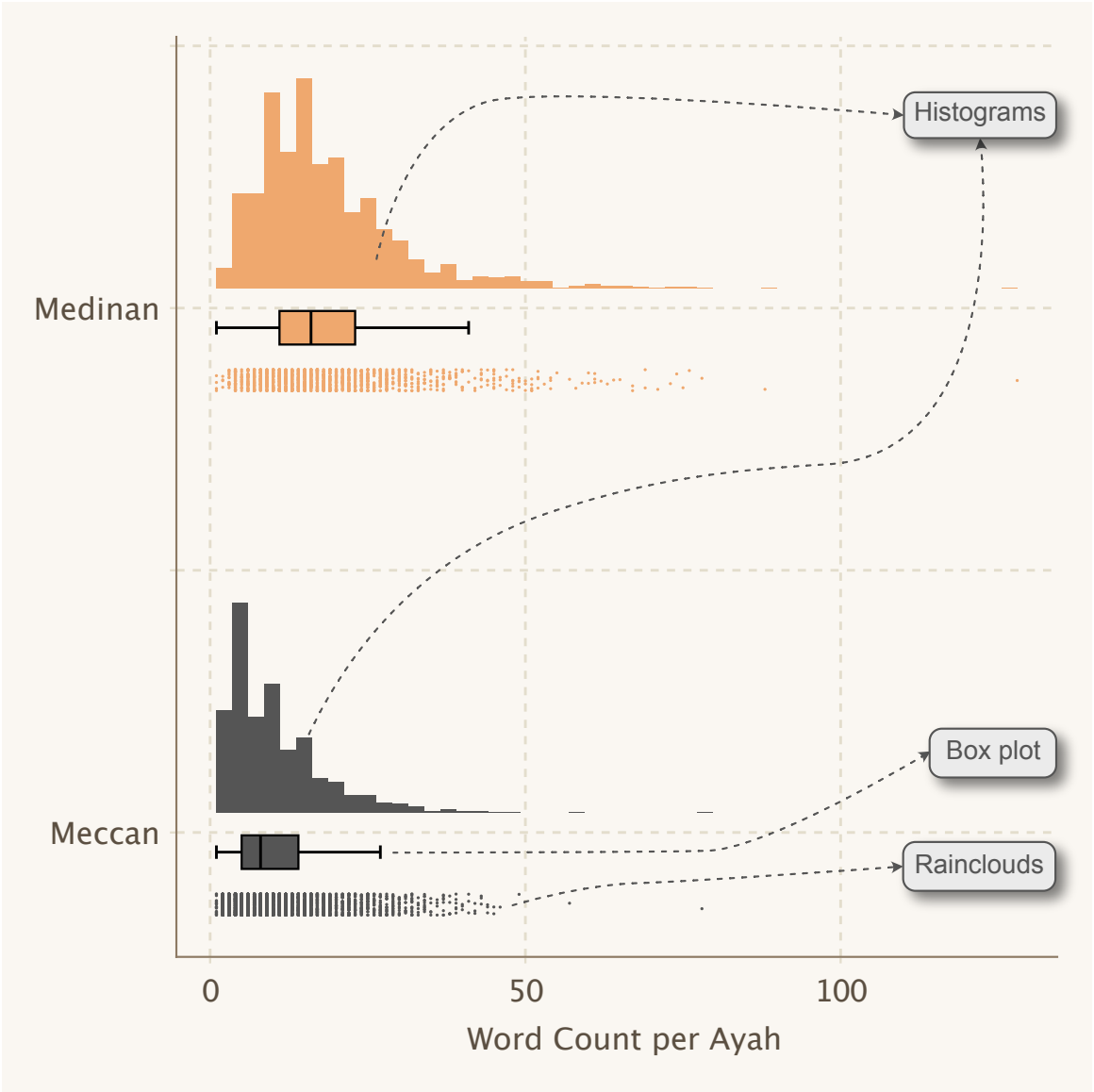


Figure 4.2: Probability density function plot of word count per *ayāt* أَيَات by revelation location, in relation to its box plot and rainclouds.

المَكِّيَّةُ, and out of these is 2602 *āyāt l-makkiyyatu* آيَات المَكِّيَّةُ with at most 10 words. Therefore, the probability is $\frac{2602}{4613} = 0.612$ or 61.2% probability. Formally, if X is the random variable of the event of observing at most 10 *kalimāt* كَلِمَات in a *āyāt l-*

makkiyyatu المَكِّيَّةُ آيَات, then

$$p(X \leq 10) = \sum_{x=0}^{10} p(X = x) \tag{4.16}$$

$$= p(X = 0) + p(X = 1) + p(X = 2) + \cdots + p(X = 10) \tag{4.17}$$

$$= 0 + \frac{24}{4613} + \frac{172}{4613} + \cdots + \frac{222}{4613} \tag{4.18}$$

$$= \frac{2602}{4613} = 0.612 \tag{4.19}$$

From Eq. 4.17, $p(X = 0)$ is the probability of observing zero *kalima* كَلِمَة in a *āyāt l-makkiyyatu* المَكِّيَّةُ آيَات, and $p(X = 1)$ is the probability of observing one *kalima* كَلِمَة in a *āyāt l-makkiyyatu* المَكِّيَّةُ آيَات, and so on. The numbers in Eq. 4.18 are the number of *āyāt l-makkiyyatu* المَكِّيَّةُ آيَات having zero, one, two, to ten *kalimāt* كَلِمَات.

⊸

4.4 Population and Sample

Statistics is a branch of science that is concerned with understanding how the data behave based on a sample—a small set of the said data. It uses statistical methodologies to understand these behavior such as probability mass/density function, and use the findings from these tools on the sampled data as a conclusion for the population—the overall data.

Figure 4.3 illustrates the relation of population and sample data. A good example of this is the political surveys on the pulse of the nation on the candidates prior to election. Private entities like PulseAsia[1] and Social Weather Station[2] (SWS) do their survey by sampling from the total population of the Philippines, hence the name survey.

The statistics computed from the surveys like the percentage of votes for particular political candidate are referred as estimates, this is because the computation was done in a sample of the population and not on the population itself. It is therefore important that for these estimates to be accurate representation of the nation's

[1] https://pulseasia.ph/

[2] https://www.sws.org.ph/swsmain/home/

opinion, it has to be representative of the population. That is, the sample shouldn't be bias and leaning to the opinions of the few only and not of the whole nation.

The importance of the sample data as illustrated below follows from the fact that it saves time and cost since interviewing 2500 compared to 100,000 is better compromise for the estimated vote percentages. Further, since these samples requires to well represent the population, the computations of the statistics are estimated using statistical models, like probability mass/density functions, and other models like linear and nonlinears discussed in Section 4.6. The next example will illustrate the idea population and sample, and how probabilistic modeling can help in understanding insights and answering more questions.

Example 4.4.1. Consider the task of sampling 250 *āyāt* آيَات from the population of *āyāt l-makkiyya*, describe the statistics of the population and the sample.
Solution: In Statistics, there are several ways to sample from data, the simplest approach is through the use of *uniform distribution*, that is, the sampling assumes that all data from the population are distributed equally, in the sense that all data points have equal chance of being selected. The other approach is through a *weighted distribution*, where a probability is assigned to each of the data points in the population, so that, the selection will be biased to those with high probability. Figure 4.4 shows the graphs or plots of the population distribution of *kalimātu l-makkiyya* كَلِمَاتُ المَكِّيَّة, which is presented as the top plot, this distribution of كَلِمَاتُ المَكِّيَّة is the same one shown in the bottom plot of Figure 4.2.

To draw 100 samples from the said population, a simple random sampling without replacement (SRSWOR) is used for selecting or drawing samples. SRSWOR works by randomly selecting sample from the population and then setting it aside as the first sample. The resulting 100 samples are plotted in the bottom plot of Figure 4.4.

As discussed above, the sample needs to be representative of the population. Figure 4.4 shows that the sampling distribution has more or less the same shape as the population. So that, the statistics are shown in Table 4.1. From the said table, it

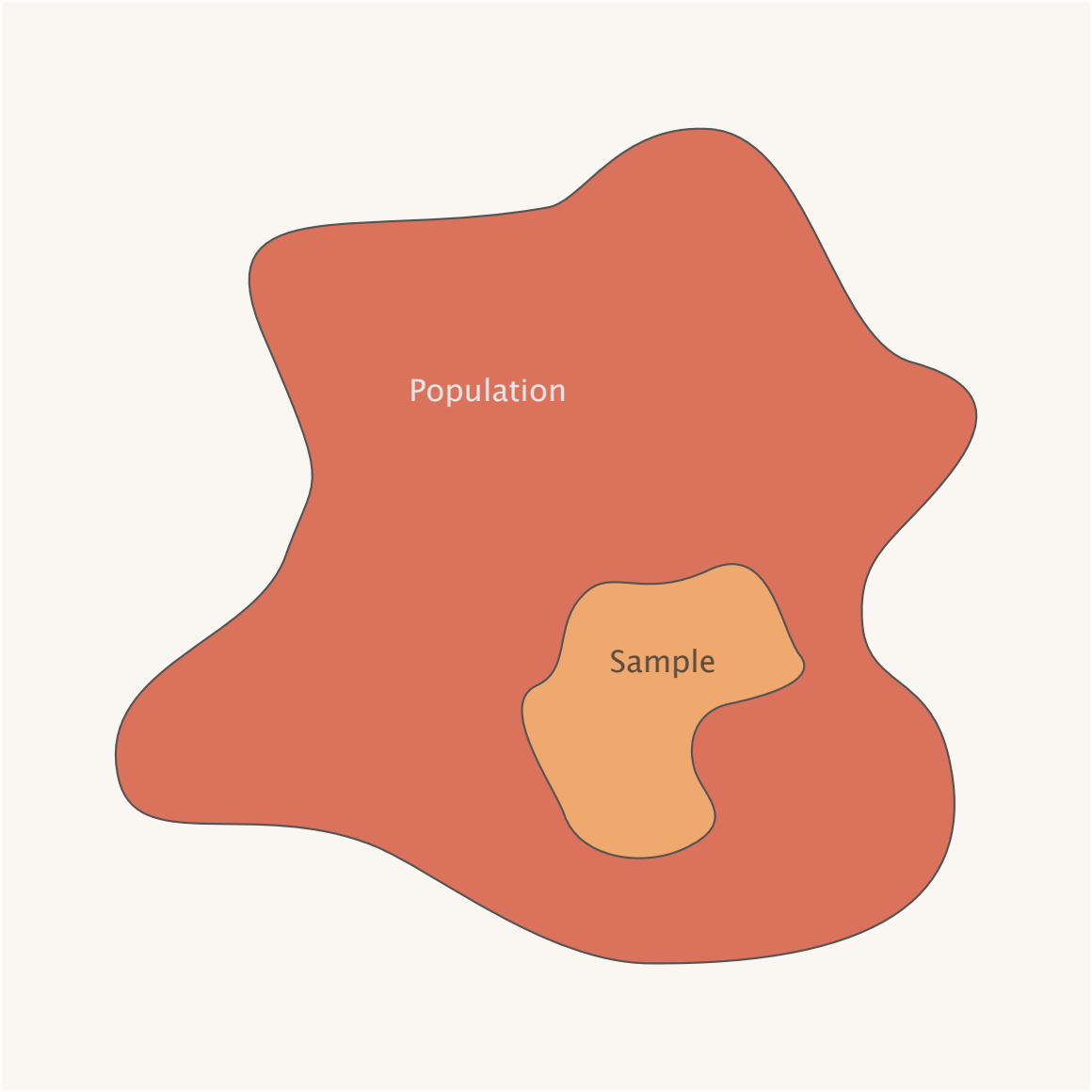


Figure 4.3: Population and sample illustration

can be seen that in terms of centrality, the both data are almost the same, for example the mean is 12.42 for the population and 10.64 for the sample. The reason why the mean in the population is much higher is due to the outlier in the population, which is seen in the extent of the tail of the Kernel Density Estimate in Figure 4.4. In fact, this is seen in the Median in Table 4.1, where the estimate are almost the same. This is because the median is not affected by the outlier. However, the variance did suffer from the outlier in the population, with 50% reduction in the sample variance, 54.15 to 48.96. The sample will less likely get the outlier as the sample since the outlier is only one observation from the 4163 total Meccan surahs.

Table 4.1: Descriptive statistics of the population and sample data of *kalimātu l-makkiyya* كَلِمَاتُ المَكِّيَّة

Data	Mean	Median	Variance	Std. Deviation
Population	10.28	8	54.15	7.36
Sample	10.64	9	48.96	7.00

The estimates got from the sample ideally are taken from a probabilistic model fitted into the sample data. This computation will be discussed in the next example.

⊸

Example 4.4.2. Consider again Ex. 4.4.1, suppose the sample data is the only data available, how will you compute the probability of getting exactly 35 *kalimātu l-sūratu l-makkiyyatu* كَلِمَاتُ السُّورَتُ المَكِّيَّةُ?

Solution: To answer this question, one might approach this using the frequency distribution as in Ex. 4.3.1. However, the use of frequency distribution from the said example is applicable since that deals with the population data, which is already the true probability, and there is no need to do some estimation. It is like try to get the average height of male Filipinos in the Philippines by doing census across the population, for such case, why would you do an estimate if you have all the census data of all heights of the Filipino, wouldn't it be easier to just do the average computation directly? This is the analogy for the frequency distribution used in Ex. 4.3.1, that is, no need to do some estimation. However, for this problem, the assumption is that only the sample is available, that is, not all of the population is available. With that said, the best solution is to do some estimation. Now, doing an estimation is possible using the samples only, that is, using the idea of frequency distribution but this time applying it on the sample. This is becaus the sample was done using a random sampling, which more or less representative of the population. However, using this approach may possess a problem, let's see what that problem will likely be by forcing the approach in Ex. 4.3.1, following the computation as in Eq. (4.16) to Eq. (4.19).

Let X again be the random variable of an event of observing 35 *kalimāt* كَلِمَات,

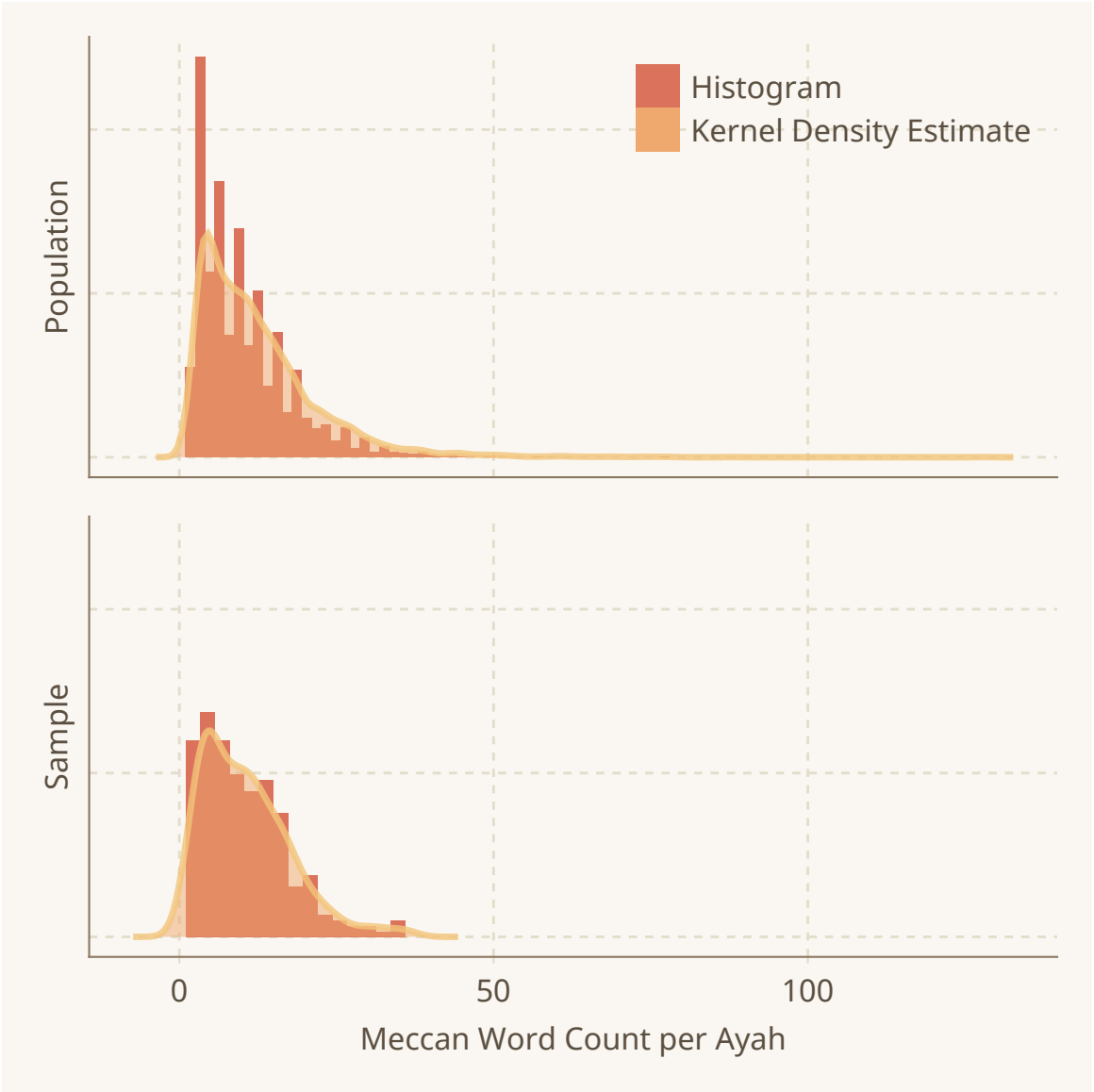


Figure 4.4: Population and sample distribution of Meccan

this time from the sample, then

$$p(X = 35) = \begin{cases} \frac{0}{250} = 0, & \text{if sample data} \\ \frac{5}{4613} = 0.0011, & \text{if population data} \end{cases} \tag{4.20}$$

From the Eq. 4.20, it shows that if we use the sample data for estimating the probability of observing 35 كَلِمَات, then the answer above is 0, meaning by estimate there is a 0 chance of observing a 35 كَلِمَات from a *āyātu l-makkiyyatu* السُّورَاتُ المَكِّيَّةُ. This conclusion is indeed misleading, since according to the population data, there are *āyātu l-makkiyyatu* آيَاتُ المَكِّيَّةُ that has 35 *kalimāt* كَلِمَات.

So, how to properly estimate this then? This is where the concept of probabilistic modeling comes in. For this problem, the data is count, and that the event of observing 10 *kalimātu l-sūratu l-makkiyyatu* كَلِمَاتُ السُّورَتُ المَكِّيَّةُ in an أَيَة is known to be best modeled by a Poisson distribution defined in Defn. 4.5.1. Ex. 4.5.1 will discuss how to solve this. ⊸•

4.5 Probability Distributions

Definition 4.5.1 (Poisson Mass Function)**.** Let X be a random variable and let $\lambda > 0$ be a parameter, then if x is the random value, then the *Poisson* mass function is given by:

$$p(X = x) = \frac{\lambda^x \exp(-\lambda)}{x!} \tag{4.21}$$

⊸○

Example 4.5.1. Consider again Ex. 4.4.1, the problem can be solved using the Poisson distribution. ⊸•

Definition 4.5.2 (Normal Density Function)**.** Let Y be a random variable and let $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}$ be the mean and variance parameters, if y is the random value, then the *Gaussian* or *Normal* density function is given below:

$$p(Y = y) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y-\mu)^2}{\sigma^2}\right\}, \quad \text{where } -\infty < y < \infty \tag{4.22}$$

⊸○

Definition 4.5.3 (Dirichlet Density Function)**.** Let $\mathbf{Y}$ be a vector random variable with a random value $\mathbf{y} := [y_1, \cdots, y_k]^{\mathrm{T}}$ and let $\boldsymbol{\alpha} := [\alpha_1, \cdots, \alpha_k]^{\mathrm{T}}$ be the parameters, then the *Dirichlet* density function is defined as

$$p(\mathbf{X} = \mathbf{x}; \boldsymbol{\alpha}) := \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{k} x_i^{\alpha_i - 1}, \tag{4.23}$$

where,

$$B(\boldsymbol{\alpha}) := \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)} \tag{4.24}$$

⊸

Definition 4.5.4 (Multinomial Mass Function)**.** Let $\mathbf{X}$ be a vector random variable with a random value $\mathbf{x} := [x_1, \cdots, x_k]^{\mathrm{T}}$ such that $x_i \geq 0, \forall i \in [1, k]$, let $\boldsymbol{q} := [q_1, \cdots, q_k]^{\mathrm{T}}$ and n be the parameters, the probability mass function of a Multinomial distribution is

$$\begin{aligned} f(x_1, \cdots, x_k; n, q_1, \cdots, q_k) :=& q(X_1 = x_1 \text{ and } \cdots \text{ and } X_k = x_k) \\ =& \begin{cases} \dfrac{n!}{x_1! \cdots x_k!} q_1^{x_1} \times \cdots \times q_k^{x_k}, & \text{when } \sum_{i=1}^{k} x_i = n \\ 0 & \text{otherwise,} \end{cases} \end{aligned} \tag{4.25}$$

⊸

4.6 Frequentist Estimation

Probability distributions are fundamental models that can be used to describe some characteristics of the data. However, combinations of variables, and accounting dynamics of the observations can lead into other types of models as well. In general, a statistical model can be written using the following formula:

$$y = f(x|\boldsymbol{\Theta}) + \varepsilon, \tag{4.26}$$

where $y \sim p(\cdot)$ and $\varepsilon \sim p(\cdot)$.

There are two main ways to estimating the parameters $\boldsymbol{\Theta}$, and that is either through Frequentist or Bayesian approach.

4.6.1 Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) is a fundamental concept in statistics that provides a method for estimating the parameters of a statistical model based on the observed data. The basic idea behind MLE is to find the parameter values that make the observed data most likely or probable under the assumed statistical model. The concept of MLE can be explained as follows:

Suppose we have a random sample of observations $X := \{x_1, x_2, ..., x_n\}$ from a probability distribution $f(x|\theta)$, where θ represents the unknown parameter(s) of the distribution. The likelihood function, denoted as $\mathcal{L}(\theta|X)$, is the joint probability density (or probability mass function for discrete distributions) of the observed data X, treated as a function of the parameter(s) θ.

$$\mathcal{L}(\theta|X) = f(x_1|\theta) \times f(x_2|\theta) \times \cdots \times f(x_n|\theta) \tag{4.27}$$

The maximum likelihood estimate (MLE) of θ, denoted as $\hat{\theta}$, is the value of θ that maximizes the likelihood function $\mathcal{L}(\theta|X)$. In other words, $\hat{\theta}$ is the value of θ that makes the observed data most likely or probable under the assumed statistical model.

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta|X) \tag{4.28}$$

In practice, it is often easier to work with the log-likelihood function, denoted as $\ell(\theta|X) = \log(\mathcal{L}(\theta|X))$, since the logarithm is a monotonic function and maximizing the log-likelihood is equivalent to maximizing the likelihood.

$$\hat{\theta} = \arg\max_{\theta} \ell(\theta|X) \tag{4.29}$$

Example 4.6.1 (Normal distribution)**.** Suppose we have a random sample $X = \{x_1, x_2, \cdots, x_n\}$ from a normal distribution with unknown mean μ and known vari-

ance σ^2. The likelihood function is:

$$\mathcal{L}(\mu|X) = \frac{1}{(\sqrt{2\pi\sigma^2})^n} \exp\left(-\sum_{\forall i}(x_i - \mu)^2/(2\sigma^2)\right) \tag{4.30}$$

Taking the log and differentiating with respect to μ, setting the derivative equal to zero, and solving for μ, we get the maximum likelihood estimate or MLE of μ:

$$\hat{\mu} = \frac{1}{n}\sum_{\forall i} x_i \tag{4.31}$$

⊸

MLE has several desirable properties, such as consistency (the MLE converges to the true parameter value as the sample size increases) and asymptotic normality (the sampling distribution of the MLE approaches a normal distribution as the sample size increases), which make it a widely used method in statistical inference and modeling.

4.6.2 Numerical Approximation

There are several ways to numerically estimate the parameters of the model using mathematical programming. The popular algorithm that is very common in Machine Learning is the *gradient descent* (GD) given in Algorithm 1. Suppose $\nabla E_{\text{in}}(\hat{\mathbf{w}}^{(r)})$ is the gradient of the cost function at the rth iteration. E_{in} is defined as the *in-sample error* or the error in the training dataset, γ is the *learning-rate* parameter of the algorithm and ν is the *precision* parameter. As an illustration, consider Example 4.6.2.

Algorithm 1 *Gradient Descent*

1: Initialize $\hat{\mathbf{w}}^{(r)}$, $r = 0$
2: **while** $\|\hat{\mathbf{w}}^{(r+1)} - \hat{\mathbf{w}}^{(r)}\| > \nu$ **do**
3: $\quad \hat{\mathbf{w}}^{(r+1)} \triangleq \hat{\mathbf{w}}^{(r)} - \gamma \nabla E_{\text{in}}(\hat{\mathbf{w}}^{(r)})$
4: $\quad r \triangleq r + 1$
5: **end while**
6: **return** $\hat{\mathbf{w}}^{(r)}$ and r.

Example 4.6.2. Suppose the loss function is given by

$$E_{\text{in}}(w) \triangleq w^4 - 3w^3 + 2. \tag{4.32}$$

The first derivative of the above equation with respect to w is given by $E'_{\text{in}}(w) = 4w^3 - 9w^2$. Let the initial guess be $\hat{w}^{(0)} = .1$ and let $\gamma = .01$ with $\nu = .00001$. Then $\nabla E_{\text{in}}(\hat{w}^{(0)}) = E'_{\text{in}}(\hat{w}^{(0)}) = -0.086$, so that $\hat{w}^{(1)} \triangleq \hat{w}^{(0)} - .01(-0.086) = 0.10086$, and $|\hat{w}^{(1)} - \hat{w}^{(0)}| = 0.00086 > \nu$. It turns out that 173 iterations are needed to satisfy the inequality, $|\hat{w}^{(r+1)} - \hat{w}^{(r)}| \not> \nu$. The plot is given in Figure **??**. ⊸•

In practice, however, there are hundreds to millions of data points that need to be summarized, so that at each iteration, the parameters are updated *after* the presentation of *all* the training examples that constitute an *epoch* — one complete presentation of the entire training dataset during the learning process (Haykin, 1998). In this setting, GD is sometimes called *batch gradient descent* (BGD).

4.6.3 Stochastic Gradient Descent

An alternative to BGD is SGD or *stochastic gradient descent*. SGD updates the parameter using only one observation for every iteration, which is a lot faster. Further, BGD is prone to local minimum since GD does so. This is guaranteed for misspecified initial value especially for high dimensional nonlinear error surface function. The stochasticity of the SGD follows from the randomization of the dataset at each epoch, and contrary to BGD, the SGD is not expected to converge to the global minimum, instead it will only stay around the global solution (*see* Algorithm 2). Example **??** illustrates the application of mathematical programming in estimating the parameters of a simple linear regression model.

4.7 Bayesian Estimation

Bayesian estimation is a statistical approach that incorporates prior knowledge or beliefs about the parameters of interest into the estimation process. It combines the prior knowledge with the observed data to obtain updated beliefs or estimates

Algorithm 2 *Stochastic Gradient Descent*

1: Initialize $\hat{\mathbf{w}}^{(r)}, r = 0$
2: **while** $\|\hat{\mathbf{w}}^{(r)} - \hat{\mathbf{w}}^{(r+1)}\| > \nu$ **do**
3: Randomize the data set $(\mathbf{x}_i, \mathbf{y}_i)$ with respect to i.
4: **for** $i \in \{1, \cdots, n\}$ **do**
5: Update the parameters

$$\begin{aligned}\hat{\mathbf{w}}^{(r)} &\triangleq \hat{\mathbf{w}}^{(r)} - \gamma \nabla \mathrm{e}(h(\mathbf{x}_i, \hat{\mathbf{w}}^{(r)}), \mathbf{y}_i) \\ &= \hat{\mathbf{w}}^{(r)} - \gamma \frac{\partial}{\partial \mathbf{w}} \left\{ \frac{1}{2} [h(\mathbf{x}_i, \hat{\mathbf{w}}^{(r)}) - \mathbf{y}_i]^2 \right\}\end{aligned}$$

6: **end for**
7: $\hat{\mathbf{w}}^{(r+1)} \triangleq \hat{\mathbf{w}}^{(r)}$
8: **end while**
9: **return** $\hat{\mathbf{w}}^{r}$ and r.

of the parameters, known as the posterior distribution. The Bayesian estimation framework is based on Bayes' theorem, which relates the conditional probabilities of events. In the context of parameter estimation, Bayes' theorem can be expressed as:

$$p(\theta|X) = \frac{p(X|\theta)p(\theta))}{p(X)} \tag{4.33}$$

where $p(\theta|X)$ is the posterior distribution, representing the updated beliefs about the parameter(s) θ after observing the data X. $p(X|\theta)$ is the likelihood function, which quantifies the probability of observing the data X given the parameter(s) θ. $p(\theta)$ is the prior distribution, representing the initial beliefs or knowledge about the parameter(s) θ before observing the data. $p(X)$ is the marginal likelihood or the probability of observing the data X, which acts as a normalizing constant.

4.7.1 Laplace's Approximation

The simplest approximation to the posterior distribution of the parameter of interest is the Laplace's approximation. The idea behind this procedure is to use a Gaussian approximator, $\mathbb{G}(x)$, such that it is centered on the mode of the target distribution,

$p(x)$. To illustrate, suppose

$$p(x) := \frac{f(x)}{Z}, \quad \text{where } Z := \int f(x)\,\mathrm{d}\,x. \tag{4.34}$$

Using basic calculus, the mode[3] of the posterior distribution, say at $x = x_{\text{MAP}}$, is achieved by taking the derivative of the objective function with respect to the x-axis, such that the gradient of the function is 0 at $x = x_{\text{MAP}}$. That is,

$$\left.\frac{\mathrm{d}\,f(x)}{\mathrm{d}\,x}\right|_{x=x_{\text{MAP}}} = 0. \tag{4.35}$$

The Gaussian distribution has the property that its logarithm is a quadratic function of the variables (?, ?). Hence, the following is an approximation of the log of the objective function using second-order Taylor series expansion centered on the mode x_{MAP},

$$\log f(x) = \log f(x_{\text{MAP}}) + \left.\frac{\mathrm{d}\log f(x)}{\mathrm{d}\,x}\right|_{x=x_{\text{MAP}}} (x - x_{\text{MAP}}) \tag{4.36}$$

$$+ \left.\frac{\mathrm{d}^2 \log f(x)}{\mathrm{d}\,x^2}\right|_{x=x_{\text{MAP}}} \frac{(x - x_{\text{MAP}})^2}{2} + O(x) \tag{4.37}$$

$$\approx \log f(x_{\text{MAP}}) + \left.\frac{\mathrm{d}^2 \log f(x)}{\mathrm{d}\,x^2}\right|_{x=x_{\text{MAP}}} \frac{(x - x_{\text{MAP}})^2}{2}. \tag{4.38}$$

Exponentiating both sides of the above equations becomes

$$f(x) \approx f(x_{\text{MAP}}) \exp\left[\left.\frac{\mathrm{d}^2 \log f(x)}{\mathrm{d}\,x^2}\right|_{x=x_{\text{MAP}}} \frac{(x - x_{\text{MAP}})^2}{2}\right], \tag{4.39}$$

so that the normalized estimator $\mathbb{G}(x)$ is given by

$$\mathbb{G}(x) = \left(-\frac{1}{2\pi}\left.\frac{\mathrm{d}^2 \log f(x)}{\mathrm{d}\,x^2}\right|_{x=x_{\text{MAP}}}\right)^{1/2} \exp\left[\left.\frac{\mathrm{d}^2 \log f(x)}{\mathrm{d}\,x^2}\right|_{x=x_{\text{MAP}}} \frac{(x - x_{\text{MAP}})^2}{2}\right]. \tag{4.40}$$

Example 4.7.1. Suppose the posterior distribution is a chi-square of the form: $p(x) :=$

[3] obtained using maximum *a posteriori* (MAP)

$\frac{x^{k-1}\exp\left(-\frac{x^2}{2}\right)}{Z}$, with k degrees of freedom where $Z := 2^{\frac{k}{2}-1}\Gamma\left(\frac{k}{2}\right)$, $x > 0$. Using Laplace, the approximator to the posterior distribution is obtained as follows:

The log-likelihood of the density function is given by

$$\ell(x) := \log p(x) = (k-1)\log x - \frac{x^2}{2} + C, \tag{4.41}$$

where C is the constant. The mode of this posterior is given by

$$\frac{\partial}{\partial x}\log p(x) = \frac{k-1}{x_{\text{MAP}}} - x_{\text{MAP}} \overset{\text{set}}{=} 0 \tag{4.42}$$

$$x_{\text{MAP}} = \sqrt{k-1}. \tag{4.43}$$

The second partial derivative of the log-likelihood with respect to x evaluated at x_{MAP} is

$$-\frac{\partial^2}{\partial x^2}\log p(x)\bigg|_{x=x_{\text{MAP}}} = 1 - \frac{1-k}{x_{\text{MAP}}^2} = 2. \tag{4.44}$$

Thus the estimate of the posterior is given by a Gaussian distribution with mean $\mu = \sqrt{k-1}$ and variance $\sigma^2 = \frac{1}{2}$. Figure **??** visualizes the Gaussian distribution as an approximator to the posterior distribution. ⊸•

Now consider the case where $\mathbf{x} \in \mathbb{R}^d$, such that $p(\mathbf{x}) := \frac{f(\mathbf{x})}{Z}$. Analogous to the univariate case, the Laplace's approximation for $p(\mathbf{x})$ is obtained by aligning the mean of the multivariate Gaussian distribution to the mode of the posterior density, denoted by $\mathbf{x}_{\text{MAP}}$. As before, the log-likelihood of $f(\mathbf{z})$ is given by the following equation

$$\log f(\mathbf{x}) \approx \log f(\mathbf{x}_{\text{MAP}}) - \frac{1}{2}(\mathbf{x}-\mathbf{x}_{\text{MAP}})^{\mathrm{T}}\mathfrak{H}(\mathbf{x}-\mathbf{x}_{\text{MAP}}), \tag{4.45}$$

where $\mathfrak{H}$ is the Hessian matrix defined by

$$\mathfrak{H} := -\frac{\partial^2}{\partial \mathbf{x}^2}\log f(\mathbf{x})\bigg|_{\mathbf{x}=\mathbf{x}_{\text{MAP}}}. \tag{4.46}$$

Exponentiating Equation (4.45) leads to the normalized approximator, $\mathbb{G}(\mathbf{x}) = \mathcal{N}_d(\mathbf{x}|\mathbf{x}_{\text{MAP}}, \mathfrak{H}^{-1})$.

4.7.2 Markov Chain Monte Carlo (MCMC)

Laplace has the advantage of being simple and easy to use. However, like any approximator, it has limitations especially on multimodal densities since it uses Gaussian as estimate to the posterior distribution. Most interesting high dimensional Bayesian models have multimodal *a posteriori*, which can't be captured through Laplace's method. To address this problem, sampling methods are instead used for approximating the *a posteriori*. These family of sampling methods are called Markov Chain Monte Carlos (MCMC) with *Metropolis-Hasting* (MH) and *Gibbs sampling* as the popular MCMCs. Further, for sophisticated MCMCs, the algorithms available are not limited to *Hamiltonian Monte Carlo* (HMC) and *Stochastic Gradient HMC*.

4.7.3 Metropolis-Hasting

The idea of the MH algorithm is to randomly walk in the support of the target density such that the random step is governed by the proposal distribution $\mathbb{G}(\cdot)$. The assumption is that the posterior distribution has no closed-form solution, but the kernel, which is the unnormalized form of the target density is easy to evaluate. This is the advantage of the Metropolis-Hasting algorithm where the *a posteriori* is not necessarily be normalized — often the difficulty in simplifying the model evidence of the Bayes' rule. Let $p(\cdot)$ be the *a posteriori*, then the Metropolis-Hasting algorithm is given in Algorithm 3.

Example 4.7.2. Consider the bivariate Gaussian distribution defined below:

$$f(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right]. \tag{4.47}$$

Suppose it has the following parameters:

$$\boldsymbol{\mu} := [10 \ -10]^{\mathrm{T}} \quad \text{and} \quad \boldsymbol{\Sigma} := \begin{bmatrix} 1.5^2 & \rho(1.5)(1.35) \\ \rho(1.5)(1.35) & 1.35^2 \end{bmatrix}$$

where $\rho := .5$; in order to draw samples from this model, let the proposal distribu-

Algorithm 3 *Metropolis-Hasting MCMC*

Initialize $\mathbf{w}_r \sim \mathbb{G}(\mathbf{w}), r = 0$
for $r \in \{1, \cdots, r_{\max}\}$ **do**
    Propose: $\mathbf{w}_{new} \sim \mathbb{G}(\mathbf{w}_{new}|\mathbf{w}_{r-1})$
    Acceptance: $\alpha(\mathbf{w}_{new}|\mathbf{w}_{r-1}) := \min\left\{1, \frac{p(\mathbf{w}_{new}|\mathbf{w}_{r-1})\mathbb{G}(\mathbf{w}_{r-1}|\mathbf{w}_{new})}{p(\mathbf{w}_{r-1}|\mathbf{w}_{new})\mathbb{G}(\mathbf{w}_{new}|\mathbf{w}_{r-1})}\right\}$
    Draw $x \sim \text{Unif}(0, 1)$
    if $x < \alpha(\mathbf{w}_{new}|\mathbf{w}_{r-1})$ **then**
        $\mathbf{w}_r := \mathbf{w}_{new}$
    else
        $\mathbf{w}_r := \mathbf{w}_{r-1}$
    end if
end for

tion defined to be the current step of the random walk plus an increment from a uniform distribution with parameters min = -5 and max = 5.

The random samples drawn by MH are not independent, this is due to the design of the algorithm where the distribution of the candidate sample depends solely on the current sample[4]. To address this problem, diagnostics are applied using methods such as *thinning*, where every ith sample is taken and the rest are discarded; or using *burn-in* where first $n^\star$ samples are discarded. So that the plot of the random samples and its kernel density are depicted in Figure **??** using 10,000 iterations. Figures **??** and **??** depict the autocorrelations.

⊸

4.7.4 Gibbs Sampling

MH is by far one of the easiest MCMC algorithm for drawing samples from *a posteriori* where direct sampling is not possible. It uses proposal distribution as drivers of the random walk in the support of the target density, which for high dimensional data, the choice of appropriate proposal function is sometimes difficult to specify. As an alternative, Gibbs sampler can be used for taking samples from the posterior distribution. The only requirement is that the joint distribution of the parameters

[4]this is the property of the Markov Chain, hence the name MCMC.

(the *a posteriori*) can be decomposed into conditional distributions of each variable conditioned on other variables. Mathematically, suppose the multivariate distribution is given by $f(\mathbf{w}|\mathscr{D})$, where $\mathbf{w} := [w_1 \; w_2 \cdots w_K]^{\mathrm{T}}$, then the Gibbs sampling algorithm is given in Algorithm 4.

Example 4.7.3. Using the same posterior distribution as in Example 4.7.2, the conditional distributions of the parameters conditioned on other parameters are also Gaussian with mean $\mu := \mu_1 + \left(\frac{\sigma_1}{\sigma_2}\right)\rho(x - \mu_2)$ and standard deviation $\sigma := \sqrt{(1-\rho^2)\sigma_1^2}$. Thus the Gibbs sampler for 10,000 iterations generates random samples shown in Figure **??**. Analogous to MH, samples obtained using Gibbs are not independent but with lower autocorrelation compared to the former. As before, the same diagnostics can be done. The autocorrelation is shown in Figures **??** and **??**, which suggest good mixing of the random samples. ⊸•

4.7.5 Bayesian Linear Regression

As an illustration of Bayesian inference to basic modeling, this section attempts to discuss the Bayesian approach to linear regression. Let $\mathscr{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ where $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$ be the pairwised dataset. Suppose the response values, $y_1, \cdots, y_n$, are independent given the parameter $\mathbf{w}$, and is distributed as $y_i \sim \mathcal{N}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i, \alpha^{-1})$, where α^{-1} is referred to as the *precision* parameter — useful for later derivation. In

Algorithm 4 *Gibbs Sampling MCMC*

Initialize $\mathbf{w}_r, r = 0$
for $r \in \{1, \cdots, r_{\max}\}$ **do**
    $w_1^{(new)} \sim f(w_1|w_2, \cdots, w_K)$
    $w_2^{(new)} \sim f(w_2|w_1^{(new)}, \cdots, w_K)$
    $w_3^{(new)} \sim f(w_3|w_1^{(new)}, w_2^{(new)}, \cdots, w_K)$
    $\vdots \qquad\qquad \vdots \qquad\qquad \vdots$
    $w_K^{(new)} \sim f(w_K|w_1^{(new)}, w_2^{(new)}, \cdots, w_{K-1}^{(new)})$
    $\mathbf{w}_r := [w_1^{(new)}, \cdots, w_K^{(new)}]^{\mathrm{T}}$
end for

Bayesian perspective, the weights are assumed to be random and are governed by some *a priori* distribution. The choice of this distribution is subjective, but choosing arbitrary *a priori* can sometimes or often result to an intractable integration, especially for interesting models. For simplicity, a conjugate prior is used for the latent weights. Specifically, assume that $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \beta^{-1}\mathbf{I})$ such that $\beta > 0$ is the hyperparameter supposed in this experiment as known value. The posterior distribution based on the Bayes' rule is given by

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{w})p(\mathbf{y}|\mathbf{w})}{p(\mathbf{y})}, \tag{4.48}$$

where $p(\mathbf{w})$ is the *a priori* distribution of the parameter, $p(\mathbf{y}|\mathbf{w})$ is the likelihood, and $p(\mathbf{y})$ is the normalizing factor. The likelihood is given by

$$p(\mathbf{y}|\mathbf{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\alpha^{-1}}} \exp\left[-\frac{\alpha(y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i)^2}{2}\right] \tag{4.49}$$

$$= \left(\frac{\alpha}{2\pi}\right)^{n/2} \exp\left[-\sum_{i=1}^{n} \frac{\alpha(y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i)^2}{2}\right]. \tag{4.50}$$

In matrix form, this can be written as

$$p(\mathbf{y}|\mathbf{w}) \propto \exp\left[-\frac{\alpha}{2}(\mathbf{y} - \mathfrak{A}\mathbf{w})^{\mathrm{T}}(\mathbf{y} - \mathfrak{A}\mathbf{w})\right] \tag{4.51}$$

where $\mathfrak{A} = \left[(\mathbf{x}_i^{\mathrm{T}})\right]$, i.e. $\mathfrak{A} \in (\mathbb{R}^n \times \mathbb{R}^d)$, this matrix is known as the *design matrix*. Given that $\mathbf{w}$ has the following prior distribution

$$p(\mathbf{w}) = \frac{1}{\sqrt{(2\pi)^d |\beta^{-1}\mathbf{I}|}} \exp\left[-\frac{1}{2}\mathbf{w}^{\mathrm{T}}\beta\mathbf{I}\mathbf{w}\right], \tag{4.52}$$

implies that the posterior has the following form:

$$p(\mathbf{w}|\mathbf{y}) \propto \exp\left[-\frac{\alpha}{2}(\mathbf{y} - \mathfrak{A}\mathbf{w})^{\mathrm{T}}(\mathbf{y} - \mathfrak{A}\mathbf{w})\right] \exp\left[-\frac{1}{2}\mathbf{w}^{\mathrm{T}}\beta\mathbf{I}\mathbf{w}\right] \tag{4.53}$$

$$= \exp\left\{-\frac{1}{2}\left[\alpha(\mathbf{y} - \mathfrak{A}\mathbf{w})^{\mathrm{T}}(\mathbf{y} - \mathfrak{A}\mathbf{w}) + \mathbf{w}^{\mathrm{T}}\beta\mathbf{I}\mathbf{w}\right]\right\}. \tag{4.54}$$

Expanding the terms in the exponent, becomes

$$\alpha \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\alpha \mathbf{w}^{\mathrm{T}}\mathfrak{A}^{\mathrm{T}}\mathbf{y} + \mathbf{w}^{\mathrm{T}}(\alpha \mathfrak{A}^{\mathrm{T}}\mathfrak{A} + \beta \mathbf{I})\mathbf{w}. \tag{4.55}$$

The next step is to complete the square of the above equation such that it resembles the inner terms of the exponential factor of the Gaussian distribution. The quadratic form of the exponential term of a $\mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}^{-1})$ is given by

$$(\mathbf{w} - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{w} - \boldsymbol{\mu}) = (\mathbf{w} - \boldsymbol{\mu})^{\mathrm{T}}(\boldsymbol{\Sigma}^{-1}\mathbf{w} - \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}) \tag{4.56}$$

$$= \mathbf{w}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{w} - 2\mathbf{w}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}. \tag{4.57}$$

The terms in Equation (4.55) are matched up with that in (4.57), so that

$$\boldsymbol{\Sigma}^{-1} = \alpha \mathfrak{A}^{\mathrm{T}}\mathfrak{A} + \beta \mathbf{I} \tag{4.58}$$

and

$$\mathbf{w}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} = \alpha \mathbf{w}^{\mathrm{T}}\mathfrak{A}^{\mathrm{T}}\mathbf{y} \tag{4.59}$$

$$\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} = \alpha \mathfrak{A}^{\mathrm{T}}\mathbf{y} \tag{4.60}$$

$$\boldsymbol{\mu} = \alpha \boldsymbol{\Sigma}\mathfrak{A}^{\mathrm{T}}\mathbf{y}. \tag{4.61}$$

Thus the *a posteriori* is a Gaussian distribution with location parameter in Equation (4.61) and scale parameter given by the inverse of Equation (4.58). The derivation above is not fully mathematical since the process of completing the square is similar to the proof of Theorem **??**.

4.8 Types of Statistical Methods

Parametric and *non-parametric* statistics are two broad categories of statistical methods, each with its own assumptions and applications. The primary difference between them lies in the underlying assumptions made about the population distribution.

4.8.1 Parametric

Parametric statistics are based on the assumption that the data follows a specific probability distribution, such as the normal distribution (also known as the Gaussian distribution). These methods rely on the estimation of parameters (such as the mean and standard deviation) from the sample data to make inferences about the population.

1. **Student's t-test** - for comparing means
2. **Analysis of Variance (ANOVA)** - for comparing means across multiple groups
3. **Pearson's correlation coefficient** - for measuring linear correlation
4. **Linear regression** - for modeling relationships between variables

4.8.2 Nonparametric

Non-parametric statistics, also known as distribution-free tests, do not make assumptions about the underlying probability distribution of the data. These methods are based on the ranks or signs of the data rather than the actual values.

1. **Mann-Whitney U test** - for comparing two independent groups
2. **Wilcoxon signed-rank test** - for comparing two related groups or paired samples
3. **Kruskal-Wallis test** - for comparing more than two independent groups
4. **Spearman's rank correlation coefficient** - for measuring monotonic relationships between variables

4.9 Types of Models

Generally, there are two types of models, the *supervised* and *unsupervised*. The other one that is not common, is the *semi-supervised*. The following sections will elaborate more.

4.9.1 Supervised

Supervised learning models are trained using labeled data, which means that each training example is paired with an output label. The goal is to learn a mapping from inputs to outputs based on the labeled training data, so the model can predict the output for new, unseen inputs. The following are examples of supervised learning models:

1. **Linear Regression** - used for predicting continuous outcomes that are not time dependent.
2. **Logistic Regression** - used for binary classification.
3. **Neural Networks** - the core model powering most of the big AI models and applications.

4.9.2 Unsupervised

Unsupervised learning models are trained using unlabeled data, which means that the data does not have output labels. The goal is to uncover the underlying structure of the data, such as grouping similar data points together or reducing the dimensionality of the data. The following are examples of unsupervised models:

1. **Hierarchical Clustering** - used for finding groups or clusters from the data.
2. **Kernel Density Estimation (KDE)** - used for estimating the probability density function of the data.
3. **Gaussian Processes** - used for regression and probabilistic classification, modeling distributions over functions.

Chapter 5

Background on Neural Networks

Detailed presentation on the theoretical results of ANN will be the focus of this chapter. Starting with the discussion on different types of *neurons* and a brief history about it. In Section 5.3, these neurons are then arranged to form networks which will be the topology of feedforward *multilayer perceptron* (MLP). Following that, are definitions of the standard notations for ANN used for the rest of this thesis. Then what follows is the optimization on the parameters. And finally, the extension to Bayesian framework for ANN.

5.1 Perceptron

From Haykin (1998), some 15 years after the publication of McCulloch and Pitts (1943) classic paper, a new approach to the pattern recognition was introduced by (Rosenblatt 1958) in his work on the *perceptron*, a novel method of supervised learning. The crowning achievement's of Rosenblatt's work was the so-called *perceptron convergence theorem*, the first proof for which was outlined by (Rosenblatt 1960b).

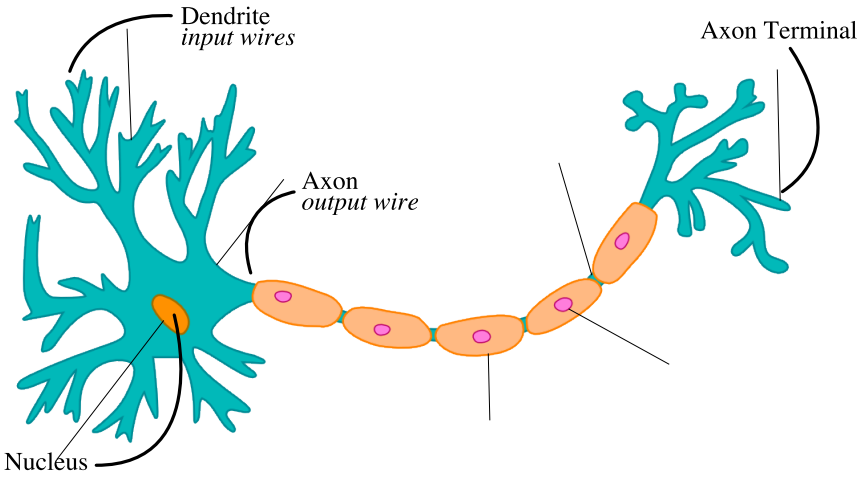


Figure 5.1: Structure of a Typical Biological Neuron.[1]

Figure 5.1 is the typical nerve cell in Biology. In relation to perceptron, the den-

[1]Original SVG image by Quasar Jarosz but colors and some labels were modified for this paper.

drite in this case takes several binary inputs, and returns a single binary output on its axon. Mathematically, let $w_1, w_2, \cdots, w_p$ be the parameters or weights, and $x_1, x_2, \cdots, x_p$ be the inputs. Then the perceptron is defined as follows:

$$y = \begin{cases} 0, & \sum_i w_i x_i \leq c \\ 1, & \sum_i w_i x_i > c \end{cases} \tag{5.1}$$

where c is a threshold set by the experimenter. The following diagram in Figure 5.2 is the schematic representation of Equation (5.1).

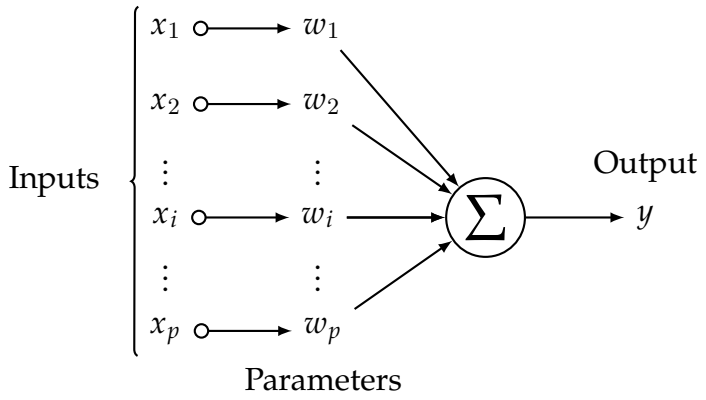


Figure 5.2: Schematic Representation of the Perceptron Neuron.

One limitation of the perceptron is its binary output, which sometimes is poor in discrimination problem, such as in imaging. To address this issue, a new artificial neuron is proposed that returns continuous values between 0 and 1, and so it can be interpreted as probabilities.

5.2 Logistic Sigmoid Neuron

The logistic sigmoid neuron or simply sigmoid neuron takes its name from the fact that it uses logistic function as its link/basis function, thus it is also known as logistic neuron. Formally, let w_0 be the intercept or the bias term and as before let $w_1, w_2, \cdots, w_p$ be the parameters, then the logistic sigmoid function is given by

$$z = \sum_i w_i x_i + w_0, \quad y = \varrho(z) = \frac{1}{1 + \exp(-z)} \tag{5.2}$$

In the proceeding section, the connections between neurons will be discussed,

and how the architecture of these networks is designed.

5.3 Topology

The basic architecture of the network comprises of the *input* and the *output* layers, between them are the *hidden* layers. These hidden layers are layers that are not observed by the experimenter, hence the name. Figure 5.3 shows the diagram of a *feedforward* neural network. It is feedforward due to the fact that the inputs are propagated onto the succeeding layers up to the output without redirecting back to the previous layer after processing. Otherwise the model is called *recurrent* neural network.

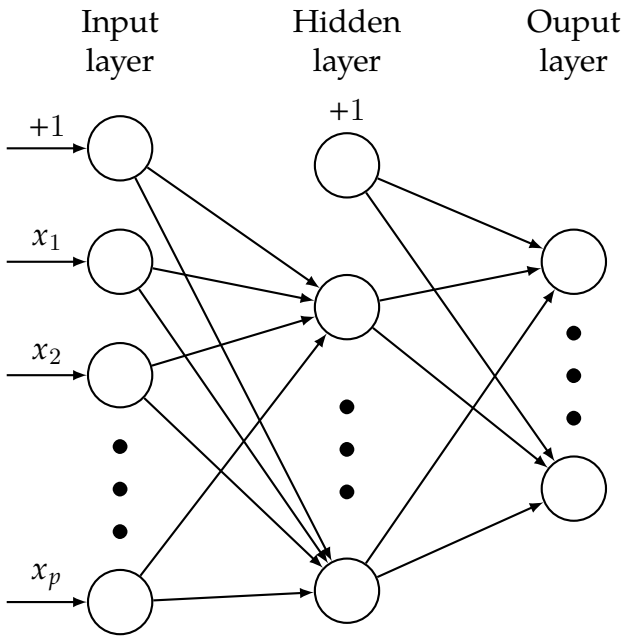


Figure 5.3: Feedforward Neural Network.

Since the input and output layers are at best observed, the hidden layers will only serve as behind-the-scene computations. And the parameter associated with each neuron has no interpretation at all, and that's one major limitation of ANN. Neural networks can have several hidden layers depending on the study, for high level of abstraction one might consider multiple layers also known as *multilayer perceptrons* (MLP), which is a misnomer since layers are made up of logistic sigmoid neurons not of perceptrons. Networks with 2 or more hidden layers are often referred to as *deep neural networks* (Nielsen, 2015), an example of this is depicted in

Figure 5.4. The unknown vector $\mathbf{w} = [(w_i)]_{i=0}^{p}$ in Equation (5.2), is the parameter vector that needs to be estimated. And this is done by maximum likelihood which is equivalent to minimization of the error function through mathematical programming using methods such as *Gradient Descent* algorithm. The succeeding section will formalize the standard notations, and then the definition of the loss function which will be the basis for choosing the best estimates of the weights, and finally the estimation.

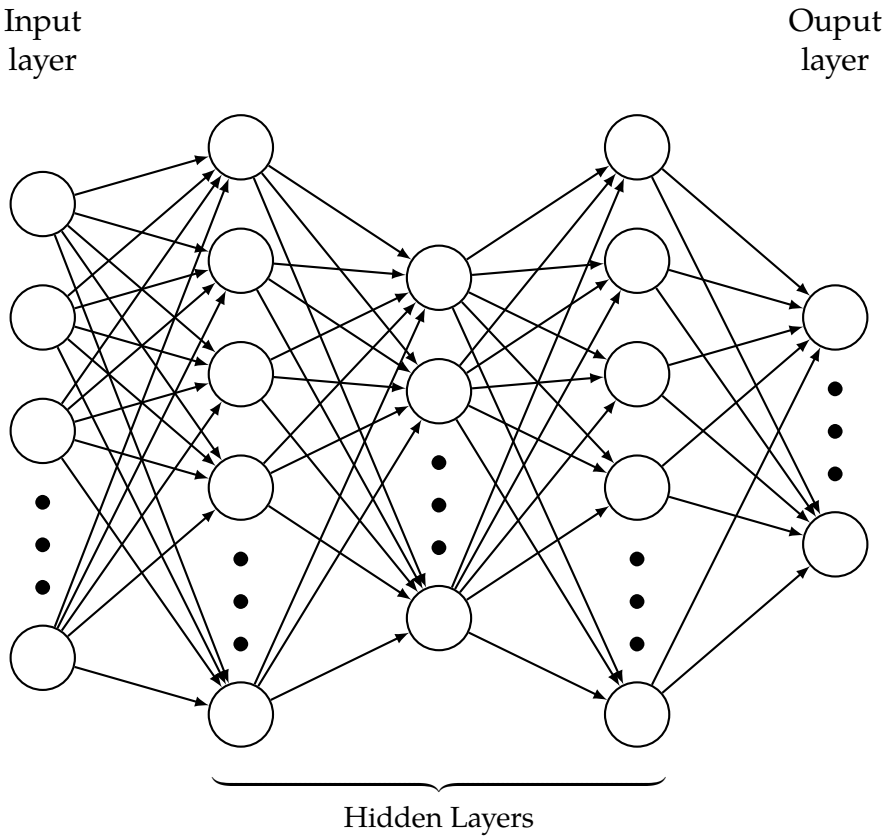


Figure 5.4: Multilayer Feedforward Neural Network.

5.3.1 Notations

The following definition will serve as the standard notation for the rest of the succeeding chapters. To start with, let $f : \mathcal{X} \to \mathcal{Y}$ be the unknown *target function* of the data points in the population, that is $y = f(\mathbf{x})$. The sample data set from this population is denoted by $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$. The learning algorithm picks $g : \mathcal{X} \to \mathcal{Y}$ as the *final hypothesis* from the hypothesis set $\mathcal{H} = \{h_v : h_v(\mathbf{x}) = \hat{y}, v = 1, 2, \cdots, V\}$

such that $g \approx f$, that is, g is the hypothesis that best approximate the target function f, thus $\hat{y} = g(\mathbf{x})$. The error in the training data set named "in-sample error" is denoted by E_{in}, while the error in the validation data set named "out-of-sample error" is denoted by E_{out}.

For ANN models, let i, j and l be the indices of the observations in the inputs, outputs and the layers containing the observations, respectively. The domain of these indices are as follows: $i \in [0, d^{(l-1)}]$, $j \in [1, d^{(l)}]$ and $l \in [1, L]$, where $d^{(l)}$ is the dimension of the outputs in the lth layer. The upper bound of the inputs is $d^{(l-1)}$, since this is the dimension of the outputs in the previous layer, $l-1$, and is now an input layer due to the foward-propagation process, in this case the next layer has $d^{(l)}$ dimension. Further, the lower bound of the input is 0 to account for the intercept term w_0. Hence the following is the standard notation of the parameters,

$$w_{ji}^{(l)} = \begin{cases} 1 \leq l \leq L & \text{layers} \\ 0 \leq i \leq d^{(l-1)} & \text{inputs} \\ 1 \leq j \leq d^{(l)} & \text{outputs} \end{cases}$$

Let $\mathbf{x}_k$ be the training input, $k = 1, 2, \cdots, K$. Since the input layer is the 0th layer with d dimension, then each input is a $d^{(0)} \times 1$ vector. That is,

$$\mathbf{x}_k = \begin{bmatrix} x_{ki}^{(0)} \\ x_{k2}^{(0)} \\ \vdots \\ x_{kd^{(0)}}^{(0)} \end{bmatrix}$$

For example in imaging, each entry in the vector corresponds to the gradient of grayscale pixel. In this setting, $h(\mathbf{x}_k, \mathbf{w})$ denotes the hypothesis or the model, where $\mathbf{w}$ is the vector of all weights. So different network achitecture relates to different model, and the final hypothesis is denoted by $g(\mathbf{x}_k, \mathbf{w})$.

The feedforward process executes as follows: the vector $\mathbf{x}_k$ will serve as an input

on the first hidden layer, whose output will then be the input on the next hidden layer, and so on until the output layer. In general,

$$x_{kj}^{(l)} = \varrho\left(z_{kj}^{(l)}\right) = \varrho\left(\left\{\mathbf{w}_j^{(l)}\right\}^{\mathrm{T}} \mathbf{x}_k^{(l-1)} + w_{j0}^{(l)}\right) = \varrho\left(\sum_{i=1}^{d^{(l-1)}} w_{ji}^{(l)} x_{ki}^{(l-1)} + w_{j0}^{(l)}\right),$$

where $\mathbf{w}_j^{(l)} = \begin{bmatrix} w_{j1}^{(1)} & w_{j2}^{(2)} & \cdots & w_{jd^{(l-1)}}^{(l)} \end{bmatrix}^{\mathrm{T}}$. This is done recursively, that is the output $x_{kj}^{(l)}$ will be the input on the next layer. For example, the following is the formula for obtaining $x_{kj}^{(l+1)}$,

$$\begin{aligned} x_{kj}^{(l+1)} = \varrho\left(z_{kj}^{(l+1)}\right) = \varrho\left(\left\{\mathbf{w}_j^{(l+1)}\right\}^{\mathrm{T}} \mathbf{x}_k^{(l)} + w_{j0}^{(l)}\right) &= \varrho\left(\sum_{i=1}^{d^{(l)}} w_{ji}^{(l+1)} x_{ki}^{(l)} + w_{j0}^{(l+1)}\right) \\ = \varrho\left\{\sum_{i=1}^{d^{(l)}} w_{ji}^{(l+1)} \left[\varrho\left(\sum_{i=1}^{d^{(l-1)}} w_{ji}^{(l)} x_{ki}^{(l-1)} + w_{j0}^{(l)}\right)\right] + w_{j0}^{(l+1)}\right\}. \end{aligned}$$

So the dimension d will vary from layer to layer. Consider Figure 5.3, the neurons can now be labelled with the standard notations as in Figure 5.5. So that $x_{kj}^{(1)}$ in the hidden layer has the following form:

$$x_{kj}^{(1)} = \varrho(z_{kj}^{(1)}) = \varrho\left(\left\{\mathbf{w}_j^{(1)}\right\}^{\mathrm{T}} \mathbf{x}_k^{(0)} + w_{j0}^{(1)}\right) = \varrho\left(\sum_{i=1}^{d^{(0)}} w_{ji}^{(1)} x_{ki}^{(0)} + w_{j0}^{(1)}\right),$$

The $x_{kj}^{(2)}$ in the output layer is computed from the following equation,

$$\begin{aligned} x_{kj}^{(2)} = \varrho(z_{kj}^{(2)}) = \varrho\left(\left\{\mathbf{w}_j^{(2)}\right\}^{\mathrm{T}} \mathbf{x}_k^{(1)} + w_{j0}^{(2)}\right) &= \varrho\left(\sum_{i=1}^{d^{(1)}} w_{ji}^{(2)} x_{ki}^{(1)} + w_{j0}^{(2)}\right) \\ = \varrho\left\{\sum_{i=1}^{d^{(1)}} w_{ji}^{(2)} \left[\varrho\left(\sum_{i=1}^{d^{(0)}} w_{ji}^{(1)} x_{ki}^{(0)} + w_{j0}^{(1)}\right)\right] + w_{j0}^{(2)}\right\}. \end{aligned}$$

The final output $\mathbf{x}_k^{(2)}$ is the value of the hypothesis, that is $h(\mathbf{x}_k, \mathbf{w}) = \mathbf{x}_k^{(L)}$, where $L = 2$.

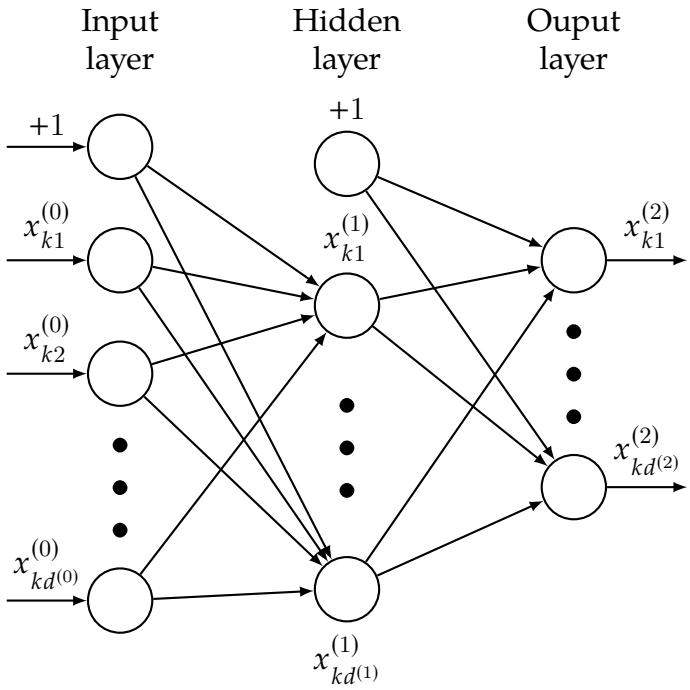


Figure 5.5: Neural Net with Standard Neuron Notation.

5.4 Network Training

Network parameter estimation is done through optimization, this is because ANN loss function has no unique stationary point. That is, it consist of local minima and often one unique global minimum.

5.4.1 Loss Function

In this section, the error functions of ANN models for regression and classification are derived from probabilistic perspective, this will be useful when extending the model to full Bayesian treatment. For regression setup, the network has at least two nodes (one for the intercept and one for the independent variable) at the input layer and one node at the output layer. The standard activation function is the identity function, i.e. $h(\mathbf{x}_k, \mathbf{w}) = z_k = \mathbf{w}^{\mathrm{T}}\mathbf{x}_k$, so there is no hidden layer, see Figure 5.6. Given the training input $\mathbf{X} = [\mathbf{x}_1^{(0)}, \mathbf{x}_2^{(0)}, \cdots, \mathbf{x}_N^{(0)}]$, where $\mathbf{x}_k^{(0)}$ is a $d^{(0)}$-dimensional feature vector whose elements are $x_{k0}^{(0)}, x_{k1}^{(0)}, \cdots, x_{kd^{(0)}}^{(0)}$ $\forall k = 1, 2, \cdots, K$, the corresponding target variable is $\mathbf{Y} = [y_1, y_2, \cdots, y_N]^{\mathrm{T}}$, and the error term ε_k is assumed to be normally distributed with mean $\mathbb{E}\varepsilon_k = 0$ and variance $\mathbb{V}\mathrm{ar}\varepsilon_k = \sigma^2$. It follows that the uncertainty around the target variable is also Gaussian distributed and is given

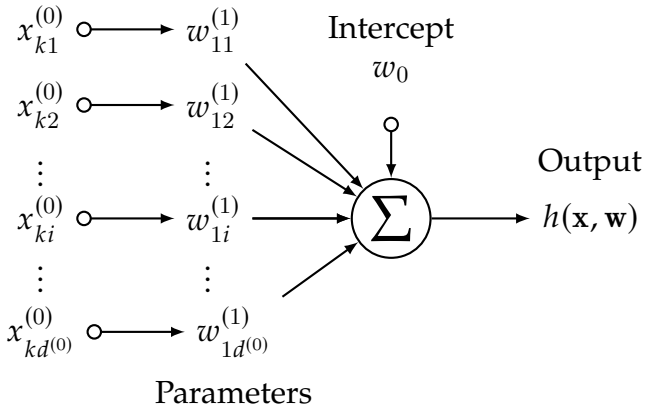


Figure 5.6: Neuron's Schematic Diagram for Linear Regression.

by

$$\begin{aligned} p(y|\mathbf{x}, \mathbf{w}, \sigma^2) &= \mathcal{N}(y|h(\mathbf{x}, \mathbf{w}), \sigma^2) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y - h(\mathbf{x}, \mathbf{w}))^2}{2\sigma^2}\right] \end{aligned} \tag{5.3}$$

If the data are independent and identically distributed, then the likelihood function of $\mathbf{y}$ is given by,

$$\begin{aligned} \mathcal{L}(\mathbf{w}, \sigma|\mathbf{y}, \mathbf{X}) = p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma) &= \prod_{k=1}^{K} \mathcal{N}(y_k|h(\mathbf{x}_k, \mathbf{w}), \sigma^2) \\ &= \frac{1}{(2\pi)^{K/2}\sigma^K} \exp\left[-\frac{1}{2\sigma^2}\sum_{k=1}^{K}(y_k - h(\mathbf{x}_k, \mathbf{w}))^2\right]. \end{aligned} \tag{5.4}$$

Taking the log-likelihood,

$$\ell(\mathbf{w}, \sigma|\mathbf{y}, \mathbf{X}) = -\frac{1}{2\sigma^2}\sum_{k=1}^{K}(y_k - h(\mathbf{x}_k, \mathbf{w}))^2 - K\log\sigma - \frac{K}{2}\log 2\pi. \tag{5.5}$$

The estimate for the weight $\mathbf{w}$, denoted by $\hat{\mathbf{w}}_{\mathrm{MLE}}$, is obtained by maximizing the log-likelihood function. To do so, partial differentiation is applied to Equation (5.5) with respect to $\mathbf{w}$, suggesting that the last two terms in the right hand side can be disregarded since these are not dependent on $\mathbf{w}$. Also, the location of the maximum log-likelihood with respect to $\mathbf{w}$ is not affected by arbitrary positive scalar multipli-

cation, so the factor $\frac{1}{\sigma^2}$ can be omitted. Equivalently, one can minimize the negative log-likelihood instead of maximizing ℓ. The reason for doing this has something to do with numerical computation, since optimizers in most statistical packages often work by minimizing the function. Thus differentiating negative of Equation (5.5) is the same as differentiating the following equation,

$$E_{\text{in}}(h) = \frac{1}{2}\sum_{k=1}^{K}(y_k - h(\mathbf{x}_k, \mathbf{w}))^2, \tag{5.6}$$

which is similar to the form of *residual sum-of-squares*. Hence the sum-of-squares error function is a consequence of maximum likelihood under the normality assumption of the uncertainty around the target variable.

Likewise, the precision parameter can be estimated using maximum likelihood, by simply differentiating Equation (5.5) with respect to σ, i.e.

$$\hat{\sigma}^2_{\text{mle}} = \frac{1}{K}\sum_{k=1}^{K}(y_k - h(\mathbf{x}_k, \mathbf{w}))^2.$$

Now consider the case for classification, in which there is a single target variable y that returns binary output, cases like when $y = 1$ corresponds to class C_1 and $y = 0$ for class C_2. The appropriate activation function is the logistic sigmoid defined in Section 5.2 that is

$$\varrho(z) = \frac{1}{1 + \exp(-z_k)},$$

implying $\varrho : \mathbb{R} \to [0, 1]$, and hence it can be interpreted as probability, this type of model has the same network architecture and neuron's schematic diagram as that in linear regression model (see Figure 5.6), but this time the activation function has unit interval range with hypothesis given by

$$\hat{y} = h(\mathbf{x}, \mathbf{w}) = \begin{cases} 1, & \varrho > c \\ \\ 0, & \varrho \leq c \end{cases},$$

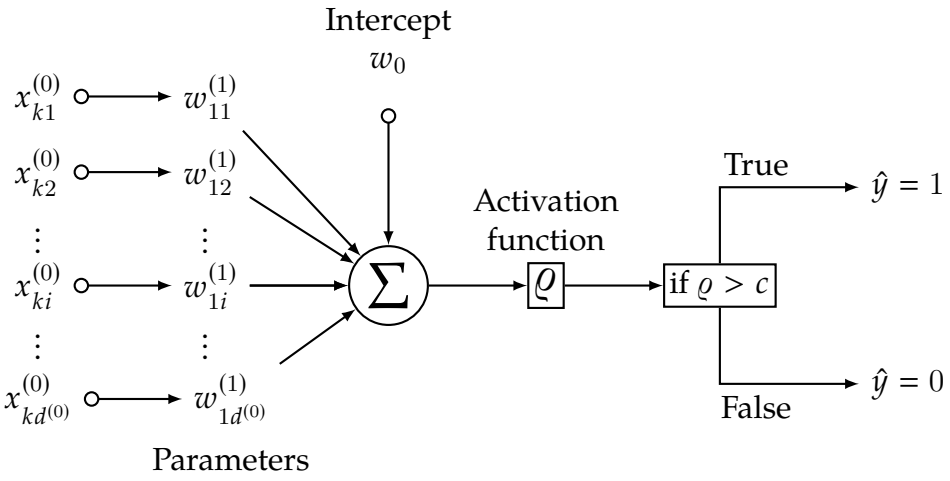


Figure 5.7: Neuron's Schematic Diagram for Logistic Regression.

where c is specified by the user, and is often chosen to be 0.5.

The probability of success, in this case $y = 1$ for C_1 is

$$p(y = 1|h(\mathbf{x}, \mathbf{w})) = h(\mathbf{x}, \mathbf{w})$$

So that the conditional distribution of the target variable y given $\mathbf{x}$ is a Bernoulli distribution with density given by

$$p(y|h(\mathbf{x}, \mathbf{w})) = \mathcal{B}(y|h(\mathbf{x}, \mathbf{w})) = h(\mathbf{x}, \mathbf{w})^y(1 - h(\mathbf{x}, \mathbf{w}))^{1-y}, \quad y \in \{0, 1\}.$$

So for sample data set $\{\mathbf{x}_k, y_k\}$ where $\mathbf{x}_k$ is a $d^{(0)}$-dimensional feature space and $y_k \in \{0, 1\}$ $\forall k = 1, \cdots, K$, if the ys are independent and identically distributed then the likelihood function is given by

$$\mathcal{L}(h(\mathbf{x}_k, \mathbf{w})|\mathbf{y}) = \prod_{k=1}^{K} h(\mathbf{x}_k, \mathbf{w})^{y_k}(1 - h(\mathbf{x}_k, \mathbf{w}))^{1-y_k},$$

and the negative log-likelihood is given by

$$\begin{aligned} E_{\text{in}}(h) &= -\ell(h(\mathbf{x}_k, \mathbf{w})|\mathbf{y}) \\ &= -\sum_{k=1}^{K} \left[y_k \log h(\mathbf{x}_k, \mathbf{w}) + (1 - y_k) \log(1 - h(\mathbf{x}_k, \mathbf{w})) \right]. \end{aligned} \tag{5.7}$$

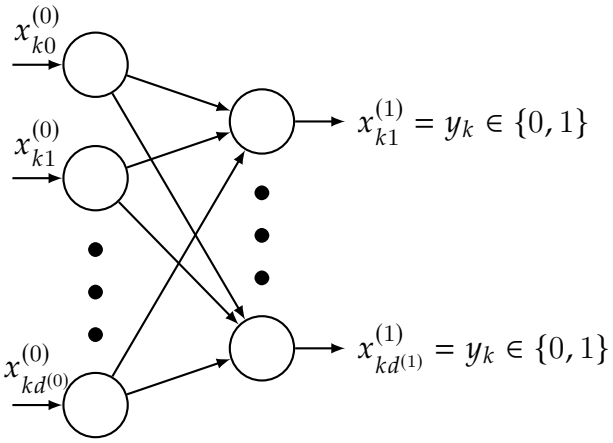


Figure 5.8: Neural Network for $d^{(1)} \geq 2$ Output Nodes

This equation is known as *cross-entropy* loss function. Extending the case to at least two separate binary classification requires $d^{(L)} \geq 2$ neurons on the output layer, each of which has a logistic sigmoid activation function. So the target variable is defined as $\mathbf{y}_k = [y_1, y_2, \cdots, y_{d^{(L)}}]^{\mathrm{T}}$, where $y_j \in \{0, 1\}\ \forall j = 1, 2, \cdots, d^{(L)}$. Assuming independence on the class labels, the joint conditional distribution of the targets given the input vector is

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \prod_{j=1}^{d^{(L)}} h(\mathbf{x}, \mathbf{w})^{y_j} \left[1 - h(\mathbf{x}, \mathbf{w})\right]^{1-y_j} .$$

Then the likelihood function is given by

$$\mathcal{L}(h(\mathbf{x}_k, \mathbf{w})|\mathbf{y}) = \prod_{k=1}^{K} \prod_{j=1}^{d^{(L)}} h(\mathbf{x}_k, \mathbf{w})^{y_{kj}} \left[1 - h(\mathbf{x}_k, \mathbf{w})\right]^{1-y_{kj}} .$$

The negative log-likelihood then becomes,

$$\begin{aligned} E_{\text{in}}(h) &= -\ell(h(\mathbf{x}_k, \mathbf{w})|\mathbf{y}) \\ &= -\sum_{k=1}^{K} \sum_{j=1}^{d^{(L)}} \left[y_{kj} \log h(\mathbf{x}_k, \mathbf{w}) + (1 - y_{kj}) \log(1 - h(\mathbf{x}_k, \mathbf{w})) \right] . \end{aligned} \tag{5.8}$$

In general, the loss function of a neural network can be written as the average

errors of the errors in the output layer across all k training inputs, i.e.

$$E_{\text{in}}(h) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{2}\sum_{j=1}^{d^{(L)}}(y_{kj} - h(\mathbf{x}_k, \mathbf{w}))^2 \tag{5.9}$$

$$= \frac{1}{K}\sum_{k=1}^{K}\varepsilon(h(\mathbf{x}_k, \mathbf{w}), y_{kj}) \tag{5.10}$$

This equation is also known as the *mean squared error*. In order to minimize this function, one has to consider numerical methods such as *gradient descent* and *stochastic gradient descent*.

5.4.2 Back-propagation Algorithm

The average error of the k training inputs defined in Section 5.4.1 is given by

$$E_{\text{in}}(\mathbf{w}) \triangleq \frac{1}{K}\sum_{k=1}^{K}\varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k). \tag{5.11}$$

The terms in this equation are errors of the training inputs. That is, for each training input there is a corresponding error that needs to be minimized. This error is the difference between the target $\mathbf{y}_k$ and the output of the hypothesis $\mathbf{h}(\mathbf{x}_k, \mathbf{w})$. To account for the iteration counter r, let $\hat{\mathbf{w}}_r$ be the notation for the estimate of $\mathbf{w}$ at the rth iteration; also let ε_r be the error at the . The inner term of Equation (5.11) can be expanded as follows:

$$\begin{aligned}\varepsilon(\mathbf{h}(\mathbf{x}_k, \hat{\mathbf{w}}_r), \mathbf{y}_k) &\triangleq \frac{1}{2}\|\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k, \hat{\mathbf{w}}_r)\|^2 = \frac{1}{2}\sum_{j=1}^{d^{(L)}}(y_{kj} - h_j(\mathbf{x}_k, \mathbf{w}))^2 \\ &= \frac{1}{2}\sum_{j=1}^{d^{(L)}}\left(y_{kj} - x_{kj}^{(L)}\right)^2 = \frac{1}{2}\sum_{j=1}^{d^{(L)}}\left[y_{kj} - \varrho\left(z_{kj}^{(L)}\right)\right]^2 \\ &= \frac{1}{2}\sum_{j=1}^{d^{(L)}}\left[y_{kj} - \varrho\left(\sum_{i=1}^{d^{(L-1)}} w_{ji}^{(L)} x_{ki}^{(L-1)} + w_{j0}^{(L)}\right)\right]^2. \end{aligned} \tag{5.12}$$

For each training input k, gradient descent algorithm is applied to the function $\varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)$. The average of all this function correspond to the loss function, so

the minimum average of the error terms will give the corresponding estimate of the parameters. For simplicity, assuming there is only one hidden layer, that is $L = 2$. Consider the following:

$$\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_{ji}^{(L)}} = \frac{\partial E_{\text{in}}(\mathbf{w})}{\partial \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)} \frac{\partial \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)}{\partial w_{ji}^{(L)}} \tag{5.13}$$

where,

$$\begin{aligned}
\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)} &= \frac{\partial}{\partial \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)} \frac{1}{K} \sum_{k=1}^{K} \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k) \\
&= \frac{1}{K} \sum_{k=1}^{K} \frac{\partial}{\partial \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)} \frac{1}{2} \sum_{j=1}^{d^{(L)}} (y_{kj} - h_j(\mathbf{x}_k, \mathbf{w}))^2 \\
&= \frac{1}{K} \sum_{k=1}^{K} \frac{\partial}{\partial \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)} \frac{1}{2} \sum_{j=1}^{d^{(L)}} \left[y_{kj} - x_{kj}^{(L)} \right]^2 \\
&= \frac{1}{K} \sum_{k=1}^{K} \sum_{j=1}^{d^{(L)}} \left[y_{kj} - x_{kj}^{(L)} \right],
\end{aligned} \tag{5.14}$$

and

$$\begin{aligned}
\frac{\partial \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)}{\partial w_{ji}^{(L)}} &= \frac{\partial}{\partial w_{ji}^{(L)}} \frac{1}{K} \sum_{k=1}^{K} \sum_{j=1}^{d^{(L)}} \left[y_{kj} - x_{kj}^{(L)} \right] \\
&= \frac{1}{K} \sum_{k=1}^{K} \left[- \left(y_{kj} - x_{kj}^{(L)} \right) \frac{\partial x_{kj}^{(L)}}{\partial w_{ji}^{(L)}} \right].
\end{aligned} \tag{5.15}$$

And note that $x_{kj}^{(L)} = \varrho\left(z_{kj}^{(L)}\right) = \varrho\left(\sum_{i=1}^{d^{(L-1)}} w_{ji}^{(L)} x_{ki}^{(L-1)} + w_{j0}^{(L)}\right)$. Hence,

$$\begin{aligned}
\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_{ji}^{(L)}} &= -\frac{1}{K} \sum_{k=1}^{K} \left(y_{kj} - x_{kj}^{(L)} \right) \frac{\partial x_{kj}^{(L)}}{\partial z_{kj}^{(L)}} \frac{\partial z_{kj}^{(L)}}{\partial w_{ji}^{(L)}} \\
&= -\frac{1}{K} \sum_{k=1}^{K} \left(y_{kj} - x_{kj}^{(L)} \right) \varrho'\left(z_{kj}^{(L)}\right) x_{ki}^{(L-1)}.
\end{aligned} \tag{5.16}$$

A gradient descent update at the $(r+1)$st iteration has the form,

$$w_{ji,r+1}^{(L)} = w_{ji,r}^{(L)} + \Delta w_{ji,r}^{(L)} = w_{ji,r}^{(L)} - \gamma \frac{\partial E_{\text{in}}(h)}{\partial w_{ji,r}^{(L)}}. \tag{5.17}$$

The $\Delta w_{ji,r}^{(L)}$ can be expressed as follows:

$$\Delta w_{ji,r}^{(L)} = -\gamma \frac{\partial E_{\text{in}}(h)}{\partial w_{ji,r}^{(L)}} = \gamma \delta_j^{(L)} x_{ki}^{(L-1)}, \tag{5.18}$$

where $\delta_j^{(L)}$ is the *local gradient* defined as

$$\begin{aligned} \delta_j^{(L)} &= -\frac{\partial E_{\text{in}}(h)}{\partial z_{kj}^{(L)}} = -\frac{\partial E_{\text{in}}(h)}{\partial x_{kj}^{(L)}} \frac{\partial x_{kj}^{(L)}}{\partial z_{kj}^{(L)}} = \frac{1}{K} \sum_{k=1}^{K} \varepsilon(h_j(\mathbf{x}_k, \mathbf{w}), y_{kj}) \varrho'\left(z_{kj}^{(L)}\right) \\ &= \frac{1}{K} \sum_{k=1}^{K} \left(y_{kj} - x_{kj}^{(L)}\right) \varrho'\left(z_{kj}^{(L)}\right). \end{aligned} \tag{5.19}$$

Thus the local gradient is the product of the error signal at the jth neuron of the Lth layer and the derivative of the associated activation function.

For weights at $(L-1)$th layer, the local gradient is then given by

$$\delta_j^{(L-1)} = -\frac{\partial E_{\text{in}}(h)}{\partial x_{kj}^{(L-1)}} \frac{\partial x_{kj}^{(L-1)}}{\partial z_{kj}^{(L-1)}} = -\frac{\partial E_{\text{in}}(h)}{\partial x_{kj}^{(L-1)}} \varrho'\left(z_{kj}^{(L-1)}\right). \tag{5.20}$$

where,

$$\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial x_{kj}^{(L-1)}} = \frac{\partial E_{\text{in}}(\mathbf{w})}{\partial \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)} \frac{\partial \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)}{\partial x_{kj}^{(L)}} \frac{\partial x_{kj}^{(L)}}{\partial z_{kj}^{(L)}} \frac{\partial z_{kj}^{(L)}}{\partial x_{kj}^{(L-1)}}. \tag{5.21}$$

And since

$$\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)} \frac{\partial \varepsilon(\mathbf{h}(\mathbf{x}_k, \mathbf{w}), \mathbf{y}_k)}{\partial x_{kj}^{(L)}} = -\frac{1}{K} \sum_{k=1}^{K} \sum_{j=1}^{d^{(L)}} \left(y_{kj} - x_{kj}^{(L)}\right) \tag{5.22}$$

then

$$\begin{aligned}\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial x_{kj}^{(L-1)}} &= -\frac{1}{K}\sum_{k=1}^{K}\sum_{j=1}^{d^{(L)}}\left(y_{kj}-x_{kj}^{(L)}\right)\frac{\partial x_{kj}^{(L)}}{\partial z_{kj}^{(L)}}\frac{\partial z_{kj}^{(L)}}{\partial x_{kj}^{(L-1)}} \\ &= -\sum_{j=1}^{d^{(L)}}\frac{1}{K}\sum_{k=1}^{K}\left(y_{kj}-x_{kj}^{(L)}\right)\varrho'\left(z_{kj}^{(L)}\right)w_{ji}^{(L)} \\ &= -\sum_{j=1}^{d^{(L)}}\delta_j^{(L)}w_{ji}^{(L)}. \end{aligned} \tag{5.23}$$

So that using Equation (5.23) in (5.20),

$$\delta_j^{(L-1)} = \varrho'\left(z_{kj}^{(L-1)}\right)\sum_{j=1}^{d^{(L)}}\delta_j^{(L)}w_{ji}^{(L)}. \tag{5.24}$$

And just to emphasize $\varrho'\left(z_{kj}^{(L-1)}\right)$ is indexed at j but at the $(L-1)$th layer, so the dimension on that layer is not necessarily the same with the Lth layer, and thus it is not included in the summation. Equations (5.24) is known as the *back-propagation equations*.

5.5 Transformers

Transformer models is a model based on the work of (Vaswani et al., 2017). These models are a type of neural network architecture designed to handle sequential data, primarily used in natural language processing (NLP) tasks. Unlike traditional recurrent neural networks (RNNs) and long short-term memory networks (LSTMs), transformers do not rely on sequential data processing. Instead, they utilize an attention mechanism that allows them to process entire sequences simultaneously, significantly improving efficiency and effectiveness.

5.5.1 Attention Mechanism

The core innovation of transformers is the self-attention mechanism, which computes attention scores to weigh the importance of different words in a sentence relative to each other.

Definition 5.5.1 (Vector Self-Attention)**.** Let $\mathbf{X} := [\mathbf{x}_1, \cdots, \mathbf{x}_N]^{\mathrm{T}}$ be the data matrix, such that $\mathbf{x}_i \in \mathbb{R}^D$ is a word embedding, then the *self-attention* is given by

$$\mathbf{y}_n := \sum_{n=1}^{N} a_{nm}\mathbf{x}_n, \tag{5.25}$$

where $0 \leq a_{nm} \leq 1$ is a attention coefficient, and that $\sum_{\forall l} a_{nm} = 1$. ⊸

The attention weights are based on the idea of *query*, *key*, and *value*, where given the an input vector $\mathbf{x}_n$, its similarity to other words $\mathbf{x}_n, n \neq m$. Here $\mathbf{x}_n$ is the query and $\mathbf{x}_m$ is the key. The similarity can be computed using *cosine similarity* (*see* discussions on this in Section 7.4.1 and Defn. 7.4.3)

Definition 5.5.2 (Attention Coefficient)**.** Let $\mathbf{X} := [\mathbf{x}_1, \cdots, \mathbf{x}_N]^{\mathrm{T}}$ be the data matrix, then the *attention coefficient* is

$$a_{nm} := \frac{\exp(\mathbf{x}_n^{\mathrm{T}}\mathbf{x}_l)}{\sum_{m'=1}^{M} \exp(\mathbf{x}_n^{\mathrm{T}}\mathbf{x}_{m'})}, \tag{5.26}$$

the softmax formula helps a_{kl} satisfy its contraints. ⊸

Definition 5.5.3 (Identity Self-Attention)**.** Let $\mathbf{y_1}, \cdots, \mathbf{y}_N$ be the *vector self-attention*, then $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N]^{\mathrm{T}}$ is the *identity self-attention* with the following formula:

$$\mathbf{Y} := \mathrm{Softmax}(\mathbf{X}\mathbf{X}^{\mathrm{T}})\mathbf{X} \tag{5.27}$$

⊸

Remark. *It is called identity self-attention since there is no learnable parameter that controls the data matrix input.*

Definition 5.5.4 (Encoded Data Matrix)**.** Let $\mathbf{V} \in \mathbb{R}^{D\times D}$, then the *encoded data matrix* is given by $\mathbf{Z} := \mathbf{X}\mathbf{V}$ ⊸

So that from Defn. 5.5.4, Eq. 5.27 becomes:

$$\mathbf{Y} = \text{Softmax}(\mathbf{X}\mathbf{V}\mathbf{V}^{\text{T}}\mathbf{X}^{\text{T}})\mathbf{X}\mathbf{V} \tag{5.28}$$

$$= \text{Softmax}(\mathbf{Z}\mathbf{Z}^{\text{T}})\mathbf{Z} \tag{5.29}$$

Further, it would be better to add learnable weights for the *query*, *key*, and *value* as well. This is formulated in the next definition.

Definition 5.5.5 (Self-Attention)**.** Let $\mathbf{W}_{(q)}, \mathbf{W}_{(k)}$, and $\mathbf{W}_{(v)}$ be the weights matrix for *query*, *key*, and *value*, respectively, such that $\mathbf{W}_{(\mathbf{i})} \in \mathbb{R}^{D \times D_i}, i \in \{q, k, v\}$, then the *self-attention* is

$$\mathbf{Y} = \text{Softmax}(\mathbf{Z}\mathbf{W}_{(q)}\mathbf{W}_{(k)}^{\mathbf{T}}\mathbf{Z}^{\mathbf{T}})\mathbf{Z}\mathbf{W}_{(v)} \tag{5.30}$$

$$\text{Softmax}(\mathbf{Q}\mathbf{K}^{\text{T}})\mathbf{V}, \tag{5.31}$$

where $\mathbf{Q} := \mathbf{Z}\mathbf{W}_{(q)}, \mathbf{K} := \mathbf{Z}\mathbf{W}_{(k)}, \mathbf{V} := \mathbf{Z}\mathbf{W}_{(v)}$. The above equation will only work if $\mathbf{W}_{(q)}$ and $\mathbf{W}_{(k)}$ have the same dimention. ⊸

Further, becaus the Softmax can suffer from having exponentially small gradients for inputs of high magnitude, the same case for tanh or logistic-sigmoid activation functions, scaling can be used to address this issue.

Definition 5.5.6 (Scaled Self-Attention)**.** From Defn. 5.5.5, the *scaled self-attention* is

$$\mathbf{Y} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) := \text{Softmax}\left[\frac{\mathbf{Q}\mathbf{K}^{\text{T}}}{\sqrt{D}}\right]\mathbf{V}, \tag{5.32}$$

where D can be D_q or D_k since $D_q = D_k$. ⊸

5.5.2 Multi-Head Attention

To enhance the model's ability to focus on different parts of the input sequence, transformers employ multi-head attention. This involves splitting the query, key, and value matrices into multiple heads, applying the attention mechanism to each head independently, and then concatenating the results.

Definition 5.5.7 (Multi-Head Attention)**.** Let $\mathbf{Y}_1, \cdots, \mathbf{Y}_H$ be the scaled self-attention such that $\mathbf{Y}_h = \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h)$, where $\mathbf{Q}_h = \mathbf{X}\mathbf{W}_h^{(q)}, \mathbf{K}_h = \mathbf{X}\mathbf{W}_h^{(k)}$, and $\mathbf{V}_h = \mathbf{X}\mathbf{V}_h^{(k)}$, then the *multi-head attention* is

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) := \text{Concat}(\mathbf{Y}_1, \cdots, \mathbf{Y}_H)\mathbf{W}^{(o)}, \tag{5.33}$$

where Concat simply combines the outputs $\mathbf{Y}_h$ into a matrix, so that, $\mathbf{W}^{(o)} \in \mathbb{R}^{H \times D_o}$.

⊸

5.5.3 Positional Encoding

Since transformers do not process input sequences in order, it require a way to incorporate the position of each word. This is achieved through positional encoding, which adds positional information to the input embeddings. Positional encodings are vectors added to the input embeddings and can be computed using sine and cosine functions of different frequencies.

Definition 5.5.8 (Positional Encoding)**.** Let $\mathbf{r}_n$ be the vector of positional encoding with components defined as:

$$r_{ni} := \begin{cases} \sin\left(\dfrac{n}{L^{i/D}}\right) & \text{if } i \text{ is even,} \\ \cos\left(\dfrac{n}{L^{(i-1)/D}}\right) & \text{if } i \text{ is odd,} \end{cases} \tag{5.34}$$

where n is the position, i is the dimension, and L is the maximum of the spectrum. For the paper of Vaswani et al. (2017) uses $L = 10000$. ⊸

5.5.4 Transformer Architecture

The transformer architecture consists of an *encoder* and a *decoder*, each comprising several layers of multi-head attention and feed-forward neural networks. The encoder processes the input sequence and generates context-aware representations, while the decoder uses these representations along with previously generated outputs to produce the final sequence.

Encoder Layer:

$$\text{EncoderLayer}(X) = \text{LayerNorm}(X + \text{MultiHead}(Q, K, V)) + \text{FFN}(X)$$

Decoder Layer:

$$\begin{aligned} \text{DecoderLayer}(Y, \text{EncoderOutput}) := \text{LayerNorm}(Y + \text{MaskedMultiHead}(Q, K, V))+ \\ \text{LayerNorm}(Y + \text{MultiHead}(Q, K, V))+ \\ \text{FFN}(Y), \end{aligned} \tag{5.35}$$

where $FFN(X)$ is a feed-forward network applied to each position separately and identically.

Transformers have revolutionized NLP by enabling efficient parallel processing and capturing long-range dependencies in text data. Their architecture, centered around self-attention and multi-head attention mechanisms, has set new benchmarks in various NLP tasks, including machine translation, text summarization, and question answering.

Chapter 6

Background on Natural Language Processing

This chapter will discuss the concept and examples of Natural Language Processing in the context of analyzing the Qur'ān.

6.1 Text Analytics

Technically, any information can be regarded as data, whether it be image, audio, video, and even texts; and in fact many more. All of these raw data, however, needs to be translated into numbers, and therefore true for texts as well. There are several ways to translate texts into numbers, the easiest way maybe is to map the letters into numbers. Like for example, a will be 1, b will be 2, and so on. This simple assignment, while easy to do, disregards a lot of information or representation of the texts. Therefore, when it comes to mapping raw data not in numeric form, the assignment has to be logical. One needs to make sure that the transformation from its raw form needs to account the characteristics of the texts. For example, any language has words that have synonyms and antonyms. So, mapping these to numbers should account these relations as well. For example, since synonyms depict the related words to a given words, their numeric form should therefore be close to each other. So that, if "beautiful" corresponds to 132, then "attractive" should have a numeric mapping that is close to 132 since the word "beautiful" and "attractive" are considered synonymous or related. On the other hand, "ugly", which is an antonym of the "beautiful", the corresponding numeric form should be far from 132 to capture the opposite of the two in number. The next section will discuss formal mathematical definition for this concept of representing the texts.

6.2 Tokenization

When it comes to analyzing natural language presented as texts, it is important that like human beings, digesting a paragraph is by understanding it at the level of word for word. This is the same with mathematical computations, the texts need to be

broken down into pieces before can be fed into the formula. Disintegrating the texts into smaller units is called *tokenization*, and that the units are called *tokens*.

Example 6.2.1. Consider the *basmala*, tokenize it into words.

Solution: Let $X =$ "بِسْمِ ٱللَّهِ ٱلرَّحْمَانِ ٱلرَّحِيمِ", if $f(\cdot; s)$ is the tokenizer function such that s is the type of delimeter, then

$$f(X; s = \text{‘’}) = [\text{بِسْمِ}, \text{ٱللَّهِ}, \text{ٱلرَّحْمَانِ}, \text{ٱلرَّحِيمِ}]^{\mathrm{T}} \tag{6.1}$$

⊸

6.3 Embeddings

Embeddings is the technical word for translating the words into numbers, it is the process of ‘embedding’ the words in a Euclidean vector space, a low-dimensional dense space. Let’s understand this piece by piece. A logical approach to translating a word into numbers would be to do a one-hot encoding, which is a process of representing a word in a sparse vector with only one non-zero entry. Let’s say the language contains only three words, ‘hi’, ‘hello’, and ‘yes’. Then, the one-hot encoding for this would be:

$$\text{hi} = [1, 0, 0] \tag{6.2}$$

$$\text{hello} = [0, 1, 0] \tag{6.3}$$

$$\text{yes} = [0, 0, 1] \tag{6.4}$$

Notice, that each of the words has unique identity in their vector encoding. However, this example only shows 3 words, what if the language contains one million words, then each words will have 999,999 zeros in their vector encoding and only one non-zero entry. If a vector contains many zeros than non-zeros, then it is refered to as sparse. For this case, it is very sparse indeed. Further, applying such methodology is not flexible, since it is tied to fix number of words, which is a problem even for Arabic due to its complex morphology. Further, sparse vectors are not good for

natural language processing (NLP), especially for doing all of the linear algebra computations. It is more preferrable to have a dense vector that also captures the semantics of the words.

Ideally, this very high dimensional vectors should be condensed into very low dimension to avoid the curse of dimensionality, a case where the vectors which supposed to be close to each other, but becomes more unique due to very high dimension. Apart from this, the embedding in the vector space should also capture the semantic proximity, that is, words that are similar semantically should be closed to each other in the embedding space. There are two approaches: *Term Frequency - Inverse Document Frequency*, and *Word Embeddings*.

6.3.1 Term Frequency - Inverse Document Frequency

The TF-IDF or Term Frequency - Inverse Document Frequency is a NLP method for embedding words. It is composed of two statistics, the TF and the IDF. The following is the formal definitions of both statistics.

Definition 6.3.1 (Term Frequency)**.** Let d be the document, and let i be a term or word in the document, then the *term frequency* of the ith term in the document d is denoted by $\mathrm{tf}(i, d)$ with the following formula: $\mathrm{tf}(i, d) = \frac{|i|}{\sum_{\forall i' \in d} |i'|}$, where i' is any other term other than i.

Definition 6.3.2 (Inverse Document Frequency)**.** Let N be the total number of documents D in the corpus, and let i be the term in d such that $d \in D$, then the *inverse document frequency* is defined by

$$\mathrm{idf}(i, d) = \log \frac{N}{|\{d : d \in D \text{ and } i \in d\}|} \tag{6.5}$$

6.3.2 Word Embeddings

TF-IDF has several limitations, like the initial discussions above, each word will have its own dimension. Therefore, the resulting embedding will be very high in dimension. Further, while TF-IDF can create pairs of words in its bag-of-words

to add a little context to the given word, it still cannot capture the full context of the sentence. Lastly, TF-IDF cannot handle out-of-vocabulary words without re-running all the computations to account the new word.

Having said all the limitaitons of the TF-IDF, *word embeddings*, which is understood as based on Neural Network became popular since it addresses all of TF-IDF limitations. First, these word embeddings are dense and lower in dimension compared to the TF-IDF embeddings. Second, these embeddings capture the semantic relationship and similarities between words. Advanced models like Bidirectional Encoder Representation from Transformer (BERT) and Generative Pre-Trained Transformers (GPT) can generate contextualized embeddings, meaning the same word can have different representations depending on its context within a sentence. Both BERT and GPT are discussed in the next chapter. Third, these embeddings have the flexibility to handle out-of-vocabulary words. Lastly, these embeddings pre-trained on large corpora can be fine-tuned for specific tasks.

In summary, while TF-IDF is a straightforward and interpretable method for measuring term importance, it falls short in capturing the semantic richness and contextual nuances of language. Neural network-based word embeddings address these limitations by providing dense, semantically rich, and context-aware representations of words, making them more powerful and versatile for a wide range of natural language processing tasks.

Chapter 7

Methodology

This chapter is organized as follows: Section **??** will discuss the concept of Topic Modeling, Section 7.3.2 will discuss the concept of Large Language Models, and finally Section 7.6 will discuss how to implement this in Julia programming language.

7.1 Morphological Analysis

This section will discuss the statistical analyses of the morphology of Qur'ān's texts. In particular, the focus would be on the parts of speech of the Qur'ān.

7.2 Rhythmic Analysis

As discussed in Chapter 1, one of the Qur'ān's characteristics is the rhythmic feature of the *aya* أَيَة in the *sūra*. This can be easily confirmed by simply learning the phonetical alphabets of Arabic, and reciting through the *aya* أَيَة of the Qur'ān. However, none has ever studied this rhythmic feature objectively from a statistical perspective. In this section, the methodology on how to analyze this is discussed. To start with, consider the following example:

Example 7.2.1. Table 7.1 shows the verses of the first chapter of the Qur'ān, *sūratu l-fātiḥa* سُورَةُ الفَاتِحَة. In this table the last two syllables are highlighted in red. In Arabic, when the sentence ends or there is a pause, the last vowel of the syllable is silent, that is, not pronounced. So that, for the first *aya* أَيَة, the last two syllables is read as *hīmi* هِيمِ, but because this is the end of the said *aya* أَيَة, then it is read as *hīm* هِيم. This is the same case with the remaining six *aya* أَيَة. Therefore, following the said rule, the *aya* أَيَة of *sūratu l-fātiḥa* سُورَةُ الفَاتِحَة are: *hīm* هِيم, *mīn* مِين, *hīm* هِيم, *dīn* دِين, *ʿin* عِن, *qīm* قِيم, and *līn* لِين. As observed, all of the ending syllables has a vowel of *i*, and this consistency is what produces the rhyme. In this paper, the rhymes across all *āyat* of the Qur'ān are studied.

Table 7.1: The verses or *āyat* آيَت of *sūratu l-fātiḥa* سُورَةُ الفَاتِحَة

Transliteration	**Verses** or *āyat* آيَت
bismi l-lahi l-raḥmāni l-rahīmi (1)	﴿١﴾ بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ
'lḥamdu lillahi rabbi 'lᶜālamīna (2)	﴿٢﴾ ٱلْحَمْدُ لِلَّهِ رَبِّ ٱلْعَٰلَمِينَ
l-raḥmāni l-rahīmi (3)	﴿٣﴾ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ
māliki yawmi l-dīni (4)	﴿٤﴾ مَٰلِكِ يَوْمِ ٱلدِّينِ
iyyāka naᶜbudu wa-iyyaka nastaᶜīnu (5)	﴿٥﴾ إِيَّاكَ نَعْبُدُ وَإِيَّاكَ نَسْتَعِينُ
ihdinā l-ṣirāṭa l-mustaqīma (6)	﴿٦﴾ إِهْدِنَا ٱلصِّرَٰطَ ٱلْمُسْتَقِيمَ
ṣirāṭa llaḏīna anᶜamta ᶜalayhim ġayri l-maġḍūbi ᶜalayhim wa-lā l-ḍāllīna (7)	صِرَٰطَ ٱلَّذِينَ أَنْعَمْتَ عَلَيْهِمْ غَيْرِ ٱلْمَغْضُوبِ عَلَيْهِمْ وَلَا ٱلضَّآلِّينَ ﴿٧﴾

→

The first method for studying the rhyme is to simply visualize this through a line chart, with the x-axis being the *aya* أَيَة number and the y-axis being the last syllables. The idea then is to understand the pattern of how the vowels switch. Figure 7.1 shows the rhythmic pattern of the last pronounced syllable of both *sūratu l-fatiḥa* سُورَةُ الفَتِحَة and *sūratu l-baqara* سُورَةُ البَقَرَة.

With the data given in Figure 7.1, we want to study if there are some interesting patterns on the switch of the rhythm, like in terms of topic, does the switch to particular rhythmic syllables imply a change in topic being discussed in the *sūra* سُورَة? Further, we also would want to model the dynamic of the changes using a statistical model. The first one is a non-dynamic model and is called Bayesian Networks model, and then the other one accounts the changes over the *aya* أَيَة numbers, and this model is called the Discrete-Time Markov Models, both discussed in the succeeding sections after the concept of Graphical models, which is the basis of the said model.

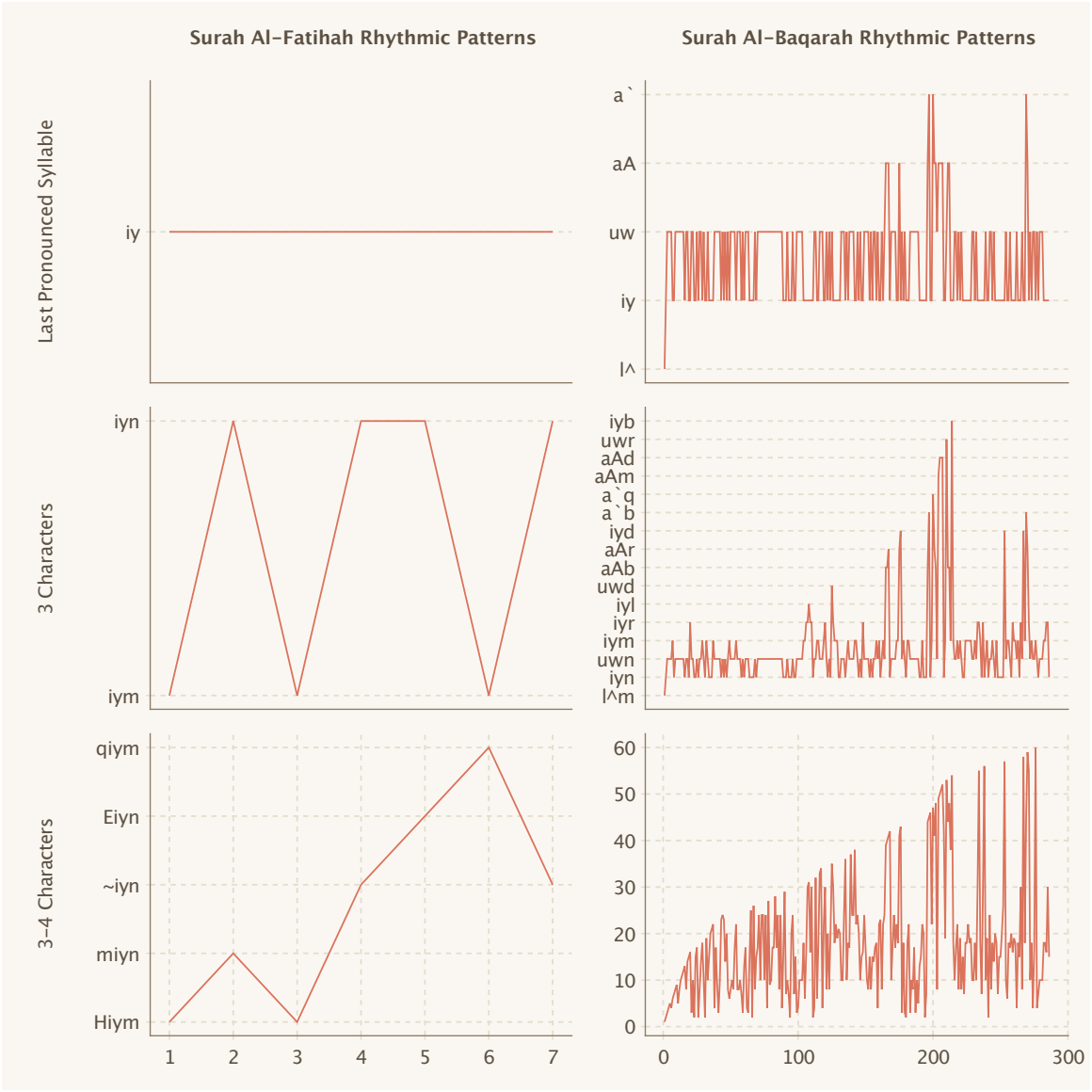
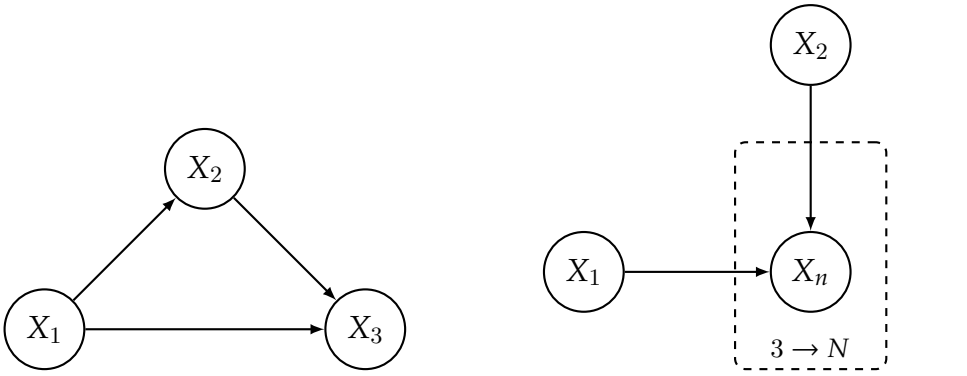


Figure 7.1: Rhythmic pattern of the last pronounced syllables of the *sūratu l-fatiḥa* سُورَةُ الفَتِحَة and *sūratu l-baqara* سُورَةُ البَقَرَة

7.2.1 Directed Probabilistic Graphical Models (PGM)

Probabilistic graphical models or simply graphical models are probability models represented graphically using nodes as random variables and segments or links as dependencies between variables. The link between the nodes is expressed as a chain rule of a conditional probability.

Example 7.2.2. Figure 7.2 (a) illustrates the connections between three variables.



(a) Directed Acyclic Graph. (b) Directed Acyclic Graph with Plate.

Figure 7.2: *Bayesian Network Representation.*

The joint probability distribution of these random variables is expressed as follows:

$$p(X_1, X_2, X_3) = p(X_1)p(X_2|X_1)p(X_3|X_1, X_2).$$

On the other hand, Figure 7.2 (b) has N events namely $X_1, X_2, \cdots, X_N$, such that the joint probability is given by

$$\begin{aligned} p(X_1, X_2, \cdots, X_N) &= p(X_1)p(X_2)p(X_3|X_1, X_2)\cdots p(X_n|X_1, X_2) \\ &= p(X_1)p(X_2)\prod_{n=3}^{N} p(X_n|X_1, X_2). \end{aligned}$$

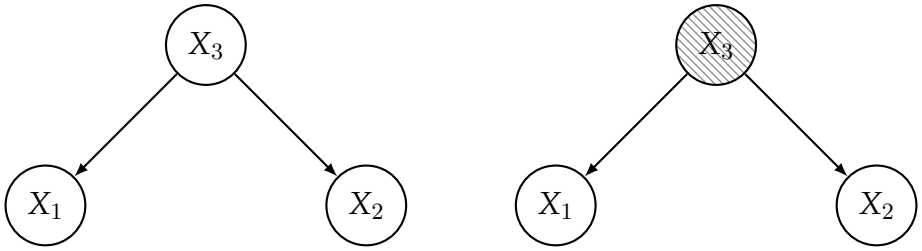
⊸•

Definition 7.2.1 (*Conditional Independence*)**.** Let X_1, X_2, and X_3 be a random variables, then X_1 and X_2 are said to be *conditionally independent* given X_3, denoted by $X_1 \perp\!\!\!\perp X_2|X_3$ if

$$p(X_1, X_2|X_3) = p(X_1)p(X_2) \tag{7.1}$$

⊸○

Example 7.2.3. This example illustrates three possible cases that are likely to occur in probabilistic graphical models. *Case 1*: In Figure 7.3 (a), X_3 is not known. The



(a) X_3 is unknown. (b) X_3 is known.

Figure 7.3: *Tail-to-Tail Node Directed Graph.*

joint probability of the graph is given by

$$p(X_1, X_2, X_3) = p(X_3)p(X_1|X_3)p(X_2|X_3).$$

It follows that $X_1 \not\perp\!\!\!\perp X_2 \mid \emptyset$, that is X_1 and X_2 are not independent since

$$\begin{aligned} p(X_1, X_2) &= \sum_{\forall x} p(X_1, X_2, X_3 = x) = \sum_{\forall x} p(X_3 = x)p(X_1|X_3 = x)p(X_2|X_3 = x) \\ &\neq p(X_1)p(X_2). \end{aligned}$$

However, if X_3 is known (see Figure 7.3 (b)), $X_1 \perp\!\!\!\perp X_2 \mid X_3$ because

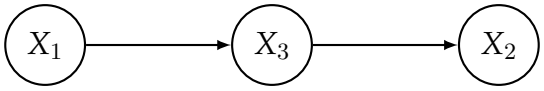
$$\begin{aligned} p(X_1, X_2|X_3) &= \frac{p(X_1, X_2, X_3)}{p(X_3)} = \frac{p(X_3)p(X_1|X_3)p(X_2|X_3)}{p(X_3)} \\ &= p(X_1|X_3)p(X_2|X_3). \end{aligned}$$

Case 2: Next consider the graph in Figure 7.4 (a), the joint probability distribution factorizes into

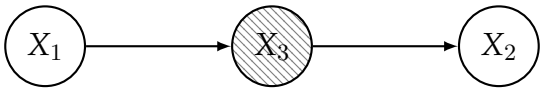
$$p(X_1, X_2, X_3) = p(X_1)p(X_3|X_1)p(X_2|X_3).$$

$X_1 \not\perp\!\!\!\perp X_2 \mid \emptyset$ since

$$p(X_1, X_2) = \sum_{\forall x} p(X_1)p(X_3 = x|X_1)p(X_2|X_3 = x) \neq p(X_1)p(X_2).$$



(a) X_3 is unknown.



(b) X_3 is known.

Figure 7.4: *Head-to-Tail Node Directed Graph.*

But if X_3 is known as in Figure 7.4 (b) then $X_1 \perp\!\!\!\perp X_2 \mid X_3$, that is

$$p(X_1, X_2|X_3) = \frac{p(X_1, X_2, X_3)}{p(X_3)} = \frac{p(X_1)p(X_3|X_1)p(X_2|X_3)}{p(X_3)}$$
$$= \frac{p(X_1)p(X_3|X_1)}{p(X_3)} p(X_2|X_3) = p(X_1|X_3)p(X_2|X_3).$$

Case 3: This case is a bit different from the preceding cases. Refer to the graphical model in Figure 7.5 (a). The joint probability distribution of this graph is given by

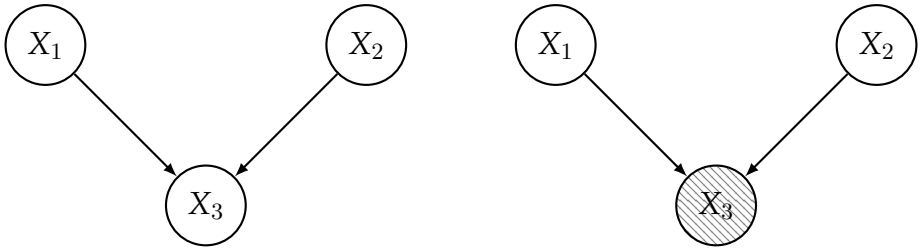
$$p(X_1, X_2, X_3) = p(X_1)p(X_2)p(X_3|X_1, X_2).$$

Previously, X_1 is not independent of X_2 if X_3 is unknown. That is not true for head-to-head relationship, however, since

$$p(X_1, X_2) = \sum_{\forall x} p(X_1)p(X_2)p(X_3 = x|X_1, X_2)$$
$$= p(X_1)p(X_2) \sum_{\forall x} p(X_3 = x|X_1, X_2) = p(X_1)p(X_2).$$

And so $X_1 \perp\!\!\!\perp X_2 \mid \emptyset$. Also $X_1 \not\perp\!\!\!\perp X_2 \mid X_3$ as shown below

$$p(X_1, X_2|X_3) = \frac{p(X_1, X_2, X_3)}{p(X_3)} = \frac{p(X_1)p(X_2)p(X_3|X_1, X_2)}{p(X_3)}$$
$$\neq p(X_1|X_3)p(X_2|X_3).$$



(a) X_3 is unknown. (b) X_3 is known.

Figure 7.5: *Head-to-Head Node Directed Graph.*

7.2.2 Markov Chain

Let $X_t,\ t = 1, 2, \cdots$ be a discrete-time stochastic process that takes values in the finite set $\mathscr{D} = \{\psi_1, \cdots, \psi_M\}$ called the *states* of the system. In particular, define the probability distribution of the initial state as

$$\eta_m := p(X_1 = \psi_m) \tag{7.2}$$

and the *transition probabilities* from state to state as

$$\upsilon_{mm^\star} := p(X_t = \psi_{m^\star} | X_{t-1} = \psi_m, \cdots, X_1 = \psi_1). \tag{7.3}$$

Definition 7.2.2 (*Markov Property*)**.** A stochastic process is said to possess a *Markov property* if the transition probability to state $m^\star$ given the history of the system, is equal to the probability of the $m^\star$th state given the immediate mth state. That is,

$$p(X_t = \psi_{m^\star} | X_{t-1} = \psi_m, \cdots, X_1 = \psi_1) := p(X_t = \psi_{m^\star} | X_{t-1} = \psi_m). \tag{7.4}$$

Suggesting that past and future states are conditionally independent given the present state. ⊸

Remark. *A process having Markov property is called* Markov process*. And the product of Equation* (7.4) *over* $t \in \{2, \cdots, \tau\}$ *is called* first-order Markov chain*.*

Example 7.2.4. Consider the Markov chain in Figure 7.6 (a) for four states. The graph is directed acyclic and so Proposition **??** is equivalent to "memorylessness"

Markov property. Suppose the history of the system from $t = 1$ to $t = 3$ are the following states: ψ_3, ψ_3, and ψ_2. The transition probability from the current state $X_3 = \psi_2$ to $X_4 = \psi_4$ is given by

$$v_{24} = p(X_4 = \psi_4 | X_3 = \psi_2).$$

The state X_3 has head-to-tail node relationship, so from Case 2 of Example 7.2.2 and Proposition **??**, $X_2 \perp\!\!\!\perp X_4 \mid X_3$ or $\psi_3 \perp\!\!\!\perp \psi_4 \mid \psi_2$. ⊸•

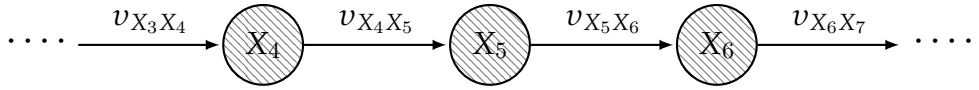
This stochastic process is also called an *observable* Markov chain since the output of the process is the set of states at each instant of time, where each state corresponds to a physical (observable) event ((?, ?)). The state diagram in Figure 7.6 (b) is a directed cyclic graph (DCG). In this setting, the Markov property is not equivalent to Proposition **??**, since d-separation will not hold. Also notice that the node is labelled as states as opposed to random variables in Figure 7.6 (a).

7.3 Thematic Analysis

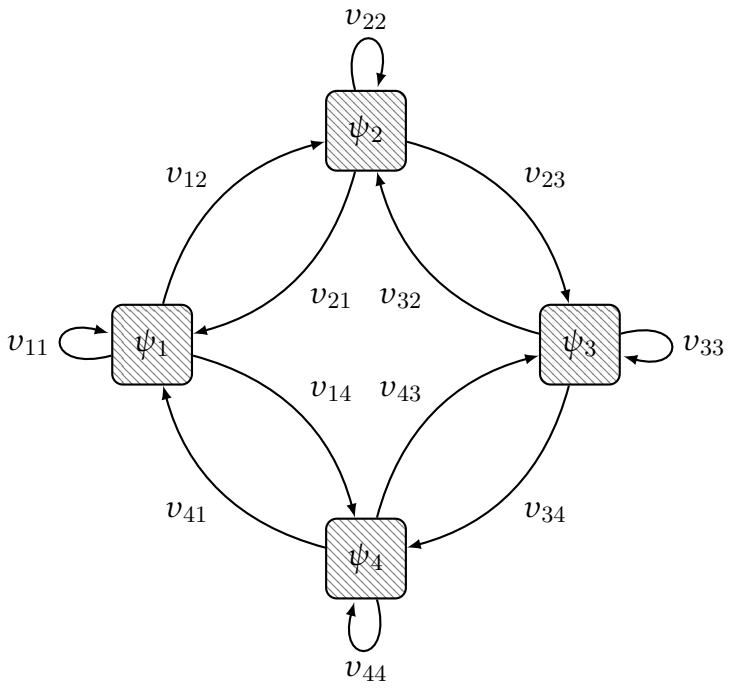
This section will discuss the methodology on how to extract the themes or topics of an input text like a *sūra* سُورَة. There are two approaches for this, using the classic statistical methodology and using the recent large language models. The popular statistical method for this is using a Bayesian model called Latent Dirichlet Allocation or LDA. On the other hand, there are two for large language model, and that is using a Bidirectional Encoder Representation from Transformers (BERT) and Generative Pre-Trained Transformer (GPT). The three approaches will be tested for generating these topics.

7.3.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a Statistical methodology that is based on Bayesian inference (Bayes, 1763; Laplace, 1986). It is a generative probabilisitic model for collection of discrete data such as text corpora (Blei, Ng, & Jordan, 2003). The main formula is defined below:



(a) First-Order Markov Chain.



(b) State Transition Diagram.

Figure 7.6: *Markov Chain Graphs.*

Definition 7.3.1 (*Latent Dirichlet Allocation*)**.** Let $\mathbf{W}, \mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}$ be the random variables, and let α and β be the hyper-parameters, then the probability of generating a document is

$$\mathbb{P}(\mathbf{W}, \mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}) = \prod_{j=1}^{m} \mathbb{P}(\boldsymbol{\theta}_j; \alpha) \prod_{i=1}^{k} \mathbb{P}(\boldsymbol{\varphi}; \beta) \prod_{t=1}^{n} \mathbb{P}(\mathbf{Z}_{j,t}|\boldsymbol{\theta}_j)\mathbb{P}(\mathbf{W}_{j,t}|\boldsymbol{\varphi}_{\mathbf{Z}_{j,t}}) \tag{7.5}$$

—∘

7.3.2 Large Language Models

Generative Artificial Intelligence or GenAI for short has been making waves on its effectiveness to generate texts, images, audio, video, etc. It has elevated humanity to a new level of capability. However, behind this amazing capabilities is that GenAI is by design a mathematical formula that are called *model*. There are several types of *models*, and one of those is the Large Language Model (LLM). The following section

will discuss what LLM is and its mathematical formulation.

7.3.3 Bidirectional Encoder Representation from Transformers

BERT or Bidirectional Encoder Representation from Transformers model is a large language model proposed by Devlin, Chang, Lee, and Toutanova (2018). From the name itself, it is based on the Transformer model architecture (*see* discussion in Section 5.5) in that it only uses the Encoder layer, and stack it together. BERT was pre-trained on large corpus of text using two unsupervised (*see* Section 4.9.2) tasks, and these are:

1. *Masked Language Modeling* (*MLM*) - tokens (*see* Section 6.2) are randomly masked in the input and trains the model to predict these masked tokens based on the surrounding context.

2. *Next Sentence Prediction* (*NSP*) - tains the model to understand the relationship between two sentences by predicting if one sentence follows the other.

After pre-training the model, BERT can then be fine-tuned on specific tasks like question answering, sentiment analysis, and more with relatively smaller datasets. With that, BERT works as follows:

1. *Input Representation* - BERT takes tokenized text as input, which includes a pair of sentences. The input is converted into tokens, added with special tokens like [CLS] (classification token at the beginning) and [SEP] (separator token between sentences).

2. *Embedding Layer* - The tokens are converted into embeddings which are the sum of token embeddings, segment embeddings, and position embeddings.

3. *Encoder Layers* - The embeddings are then passed through multiple layers of bidirectional Transformer encoders (*see* Section 5.5), which apply self-attention mechanisms to generate contextualized representations for each token.

4. *Output* - The final hidden states from the encoder layers are used for different tasks:

- The [CLS] token's representation can be used for classification tasks.
- The representations of other tokens can be used for tasks like named entity recognition (NER) or question answering.

There are several applications of BERT model, but for this paper it will be used for Topic Modeling and Text Summarization of the Qur'ān. In particular, CL-AraBERT model by will be used for extracting embeddings of the Qur'ānic words for further analysis.

7.3.4 Generative Pre-Trained Transformer

GPT or Generative Pre-Trained Transformer is another large language model proposed by (Radford, Narasimhan, Salimans, Sutskever, et al., 2018). From the name itself and like BERT, GPT is based on the Transformer model (Vaswani et al., 2017), *see* Section 5.5. Unlike BERT though, GPT uses the decoder layer of the Transformer model and stacks it multiple times. This is the model that is powering the ChatGPT[1] of OpenAI and also Claude AI[2] of Anthropic[3].

GPT models like those powering ChatGPT were pre-trained on large corpora by going through the sequence of the texts in *unidirection*, which is contrary to the *bidirectional* approach of BERT model. As such, the GPT models excel in generating text and performing tasks that require producing coherent sequences of words, in applications like text completion and creative writing. Whereas BERT is technically effective for tasks requiring deep contextual understanding such as text classification and named-entity recognition.

For this paper, the

7.4 Concentrism Statistical Formulation

One of the specific items for the second objective of this paper is on the theory of concentrism, and how can this be formulated statistically, and what are the insights that can be extracted. The idea of the theory of concentrism is that a given texts with

[1] https://chat.openai.com/
[2] https://claude.ai/
[3] https://anthropic.com/

define partition follows a ring or concentric structure, which is a literary form where the text is organized in such a way that it mirrors itself around a central point. This means that the beginning and ending sections correspond to each other, moving inward until the center of the text, which often contains the main message or theme. This particular pattern was observed by linguistic experts that it got documented in a book by Farrin (2014). This theory can be defined mathematically as follows:

Definition 7.4.1 (*Concentric*)**.** Let $\mathscr{D}$ be a collection of texts, then $\mathscr{D}$ is said to have a *concentric* or *ring* structure if and only if $\exists\ \mathscr{A}_i \subseteq \mathscr{D}, \mathscr{C} \subseteq \mathscr{D}$, and $\mathscr{A}_i^* \subseteq \mathscr{D}, i \in \mathbb{N}_1$; and that these sets are arranged as follows in $\mathscr{D}$: $\mathscr{A}_1, \cdots, \mathscr{A}_n, \mathscr{C}, \mathscr{A}_1^*, \cdots, \mathscr{A}_n^*$, such that $\mathscr{A}_i^*$ mirrors $\mathscr{A}_i$ in semantic, and that $\mathscr{C}$ is the center texts of the document $\mathscr{D}$ that is not related to both $\mathscr{A}_i$ and $\mathscr{A}_i^*$. ⊸

The underlined words above will be used in the next section, because mathematically it begs further definition on what we mean by "mirrors" and "not related" mathematically. This will be defined in the next section. Now, another pattern that was observed by Farrin (2014) as chiasmus, which is basically *ring* structure but the second half of the ring after the center is the "complement" or "reversal" of the first half of the document before the center. The following is its mathematical definition.

Definition 7.4.2 (*Chiasmus*)**.** Let $\mathscr{D}$ be a collection of texts, then $\mathscr{D}$ is said to have a *concentric* or *ring* structure if and only if $\exists\ \mathscr{A}_i \subseteq \mathscr{D}, \mathscr{C} \subseteq \mathscr{D}$, and $\mathscr{A}_i^* \subseteq \mathscr{D}, i \in \mathbb{N}_{geq1}$; and that these sets are arranged as follows in $\mathscr{D}$: $\mathscr{A}_1, \cdots, \mathscr{A}_n, \mathscr{C}, \mathscr{A}_1^c, \cdots, \mathscr{A}_n^c$, such that $\mathscr{A}_i^c$ is the complement or reversal of $\mathscr{A}_i$ in semantic, and that $\mathscr{C}$ is the center texts of the document $\mathscr{D}$ that is not related to both $\mathscr{A}_i$ and $\mathscr{A}_i^c$. ⊸

There are other structures that can be observed in the Qur'ān, like *parallelism* where themes are repeated in other *sūra* سُورَة; and *segment structure*, where a particular segment starts and ends with similar phrases or themes, creating a bracket around the content; but, these other structures are not studied in this paper. Apart from this, there is also the "mathematical patterns" of the Qur'ān which has been extensively studied by Khalifa (1981), but this is also not studied in this paper.

7.4.1 Cosine Similarity

From Defn. 7.4.1 and 7.4.2, there are key words that are still vagued in terms of its mathematical meaning, and these were ”mirrors,” ”not related,” and ”complemented” or ”reversal.” Well, semantically these words refer to measurement, particularly, a distance measurement. So that, ”mirrors” would mean related meaning the distance in terms of measurement should be relatively close as opposed to ”not related” or ”complemented” or ”reversal”. So, how to measure this then?

The answer to the above question is by measuring the distance of their word embeddings. These embeddings as discussed in Section 7.3.3 will be extracted from BERT models. Using these embeddings a similarity or distance measurement can be used, and the common formula for this is the *cosine similarity* defined below.

Definition 7.4.3 (*Cosine Similarity*)**.** Let $\mathbf{u} := [u_1, \cdots, u_n]^{\mathrm{T}}$ and $\mathbf{v} := [v_1, \cdots, v_n]^{\mathrm{T}}$, $n \in \mathbb{N}_{\geq 1}$ be word embedding vectors such that $\theta_{\mathbf{u},\mathbf{v}}$ is the angle between $\mathbf{u}$ and $\mathbf{v}$, then the cosine similarity of the given angle θ is

$$\cos(\theta_{\mathbf{u};\mathbf{v}}) := \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||\,||\mathbf{v}||}. \tag{7.6}$$

⊸

Therefore, for this study, if θ_1 is the angle between ”related” *āyat* آيَت embeddings, and θ_2 is the angle between ”not related” *āyat* آيَت embeddings, then it should be expected that $\cos(\theta_1) \leq \cos(\theta_2)$. Further, if θ_3 is the angle of ”complemented” or ”reversal” *āyat* آيَت embeddings, then maybe $\cos(\theta_1) \leq \cos(\theta_3)$ since it is not clear yet how disparate is the ”related” distance to ”complement” or ”reversal” distance, and this is especially true for the relation of θ_2 to θ_3 as it cannot be determined up front, and may only be observed from the data, which will be discussed in the next chapter.

7.4.2 Bayesian Optimization

From Defn. 7.4.1 and 7.4.2, it both states that, a document $\mathscr{D}$ may only be considered *concentric* or *chiasmus* if and only if "there exist", denoted by the symbol $\exists$. Indeed, subset of texts $\mathscr{A}_i$s, $\mathscr{C}$, and $\mathscr{A}_i^*$ or $\mathscr{A}_i^{\mathrm{c}}$ are all determined manually by the investigator. In this study, we propose to automate the determination of these subsets of texts of document $\mathscr{D}$, and this is through the use of an optimization algorithm called Bayesian Optimization. The idea behind Bayesian optimization is that it approximates the objective function with a *surrogate* function, and using this surrogate to find the global maximum or minimum of the objective function. Furthermore, it does this by fitting the surrogate function to the objective function in as few input points or iteration as possible. As such, an *acquisition* function is needed for choosing smartly the next input points to be used for fitting the surrogate. Using these components, a Bayes' Theorem is used in fitting the surrogate.

To do this, a *global objective function* must be defined, for this study, the cosine similarity discussed in the previous section will be used as the said cost function that will either be maximized or minimized. The core model for this optimization is the Gaussian Process defined below.

Surrogate Function

The surrogate function aims to approximate the objective function. In Bayesian, optimization, the surrogate function is defined by the Gaussian process (GP).

Definition 7.4.4 (*Gaussian Process*)**.** Let $Y_1, Y_2, \cdots, Y_T$ be sequence of random variables such that $Y_t \in \mathbf{Y}$, then the sequence is a *Gaussian process* (GP) if and only if $\mathbf{Y} \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. ⊸

Definition 7.4.5 (*Parameter Space*)**.** Let $\mathbf{s}_i \in \mathscr{D}$ be a word embedding of an *aya* أَيَة in document $\mathscr{D}$, which can be a *sūra* سُورَة, group of *suwar* سُوَر, or group of *āyat* آيَت within a *sūra* سُورَة. Further, suppose $\mathscr{A}_i, \mathscr{C}, \mathscr{A}_i^* \subseteq \mathscr{D}$ such that $\mathscr{A}_i := \{\mathbf{s}_{i,1}, \cdots, \mathbf{s}_{i,n}\}$, $\mathscr{C} := \{\mathbf{s}_{n+1}, \cdots, \mathbf{s}_{n+c}\}$, and $\mathscr{A}_i^* := \{\mathbf{s}_{i,(n+c+1)}, \cdots, \mathbf{s}_{i,(n+c+n)}\}$, then the *parameters* to be optimized are assigned to $\mathbf{v} := [n, c]^{\mathrm{T}}$, and since $n, c, m \in \mathbb{N}_{\geq 2}$, then the *parame-*

ter space is $\mathscr{P} := \mathbb{N}_{\geq 2} \times \mathbb{N}_{\geq 2} \times \mathbb{N}_{\geq 2} = (\mathbb{N}_{\geq 2})^3$, so that $\mathbf{v} \in \mathscr{P}$. ⊸

Definition 7.4.6 (*Semi-Circle Cost Function*)**.** From Defn. 7.4.5, let r_1 to denote that $\mathscr{A}_i$ is "related" to $\mathscr{A}_i^*$, and r_2 to denote that $\mathscr{A}_i$ is "reversal" of $\mathscr{A}_i^*$, such that $\eta := \{r_1, r_2\}$, then the *semi-circle* cost function between $\mathscr{A}_i$ and $\mathscr{A}_i^*$ is defined as:

$$\gamma_a(\mathbf{v}; \mathscr{A}_i, \mathscr{A}_i^*, \eta) := \begin{cases} \frac{1}{n} \sum_{k=1}^{n} \cos\left(\theta_{\mathbf{s}_{i,k};\mathbf{s}_{i,n+l}}\right), & \text{if } \eta = r_1 \\ 1 - \frac{1}{n} \sum_{k=1}^{n} \cos\left(\theta_{\mathbf{s}_{i,k};\mathbf{s}_{i,n+l}}\right), & \text{if } \eta = r_2 \end{cases} \tag{7.7}$$

⊸

Definition 7.4.7 (*Center Cost Function*)**.** From Defn. 7.4.5, the *center* cost function is defined as:

$$\gamma_b(\mathbf{v}; \mathscr{A}_i, \mathscr{C}, \mathscr{A}_i^*) := 2 - \frac{1}{nc} \sum_{k=1}^{n} \sum_{l=1}^{c} \cos\left(\theta_{\mathbf{s}_{i,k};\mathbf{s}_{n+l}}\right) - \frac{1}{nc} \sum_{k=1}^{n} \sum_{l=1}^{c} \cos\left(\theta_{\mathbf{s}_{n+l};\mathbf{s}^*_{i,n+c+k}}\right). \tag{7.8}$$

⊸

Definition 7.4.8 (*Global Cost Function*)**.** From Defn. 7.4.5-7.4.7, the *global* cost function is defined as:

$$\gamma_g(\mathbf{v}; \mathscr{A}_i, \mathscr{C}, \mathscr{A}_i^*, \eta) := \gamma_a(\mathbf{v}; \mathscr{A}_i, \mathscr{A}_i^*, \eta) + \gamma_b(\mathbf{v}; \mathscr{A}_i, \mathscr{C}, \mathscr{A}_i^*) \tag{7.9}$$

⊸

Definition 7.4.9 (*Optimal Parameters*)**.** From Defn. 7.4.8, the optimal parameters would be:

$$\hat{\mathbf{v}} := \arg\min_{n,c} \gamma_g(\mathbf{v}; \mathscr{A}_i, \mathscr{C}, \mathscr{A}_i^*, \eta), \tag{7.10}$$

⊸

Proposition 7.4.1. *Let $\mathscr{P}$ be the parameter space and suppose $\gamma(\mathbf{v}) \overset{\text{iid}}{\sim} \mathcal{N}(m(\mathbf{v}), k(\mathbf{v}, \mathbf{v}))$, $\forall \mathbf{v} \in \mathscr{P}$, then for any $\boldsymbol{\delta} := [\gamma(\mathbf{v}_1), \cdots, \gamma(\mathbf{v}_n)]^\top$, $\boldsymbol{\delta} \overset{\text{iid}}{\sim} \mathcal{N}_n(\mathbf{m}, \mathbf{K})$, where*

$$\mathbf{m} := [m(\mathbf{v}_1), \cdots, m(\mathbf{v}_n)]^\top \tag{7.11}$$

and

$$\mathbf{K} := \begin{bmatrix} k(\mathbf{v}_1, \mathbf{v}_1) & \cdots & k(\mathbf{v}_1, \mathbf{v}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{v}_n, \mathbf{v}_1) & \cdots & k(\mathbf{v}_n, \mathbf{v}_n) \end{bmatrix} \tag{7.12}$$

Proof. The proof follows from the proof of Theorem 1.2.9 of Muirhead (2005). □

Remark. *From Proposition 7.4.1 and Definition 7.4.4, $\boldsymbol{\delta}$ is a GP.*

Proposition 7.4.2. *From Proposition 7.4.1, suppose $\boldsymbol{\delta}_* := [\gamma(\mathbf{v}_{n+1}), \cdots, \gamma(\mathbf{v}_{n+p})]^\top$, such that $\boldsymbol{\delta}_* \overset{\text{iid}}{\sim} \mathcal{N}_p(\mathbf{m}_*, \mathbf{K}_*)$ then*

$$\begin{bmatrix} \boldsymbol{\delta} \\ \boldsymbol{\delta}_* \end{bmatrix} \overset{\text{iid}}{\sim} \mathcal{N}_{n+p}\left(\begin{bmatrix} \mathbf{m} \\ \mathbf{m}_* \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix} \right) \tag{7.13}$$

Proof. Let $\mathbf{u} := [\boldsymbol{\delta}, \boldsymbol{\delta}_*]^\top$, then $\mathbf{u} = [\gamma(\mathbf{v}_1), \cdots, \gamma(\mathbf{v}_{n+p})]^\top$. Further, since $\gamma(\mathbf{v}_i) \overset{\text{iid}}{\sim} \mathcal{N}(m(\mathbf{v}_i), k(\mathbf{v}_i, \mathbf{v}_i)), \forall i \in \mathbb{N}_{\leq n+p}$, then the joint distribution of the $\gamma(\mathbf{v}_i)$s, i.e. $\Pr(\mathbf{u})$, follows from the proof of Theorem 1.2.9 of Muirhead (2005). □

Corollary 7.4.1. *From Proposition 7.4.2, let $\mathbf{m}$ and $\mathbf{m}_*$ be zero vectors, then the following conditional distribution is true:*

$$\boldsymbol{\delta}_* \mid \boldsymbol{\delta} \overset{\text{iid}}{\sim} \mathcal{N}_p(\mathbf{K}_*^\top \mathbf{K}^{-1} \boldsymbol{\delta}, \mathbf{K}_{**} - \mathbf{K}_*^\top \mathbf{K}^{-1} \mathbf{K}_*). \tag{7.14}$$

Proof. The proof follows by reversing the condition from $\mathbf{X}_1 \mid \mathbf{X}_2$ to $\mathbf{X}_2 \mid \mathbf{X}_1$ of the proof of Theorem 1.2.11 of (Muirhead, 2005). □

Definition 7.4.10 (*Lower Confidence Bound*)**.** Let $m(\mathbf{v})$ be the mean of the GP, then the Lower Confidence Bound (LCB) notated as ξ is given by

$$\xi(\mathbf{v} \mid \zeta) := m(\mathbf{v}) - \zeta k(\mathbf{v}, \mathbf{v}), \tag{7.15}$$

where ζ is the balancing factor. ⊸

The *global cost* computed from the global cost function in Defn. 7.4.8 is approximated by the Gaussian process (the surrogate function) defined in Defn. 7.4.4

through Proposition 7.4.1. So that, for any new observed global cost computed from the new parameter input $\mathbf{v}$, chosen by the acquisition function defined in Defn. 7.4.10, the joint distribution with the previous global cost is given in Proposition 7.4.2. Therefore, computing for the conditional distribution of the new global costs conditioned on previous global cost is given in Corollary 7.4.1. The process of acquiring new parameter candidate and computing the global cost is done iteratively. The algorithm converges until there is no significant changes on the new parameter candidate relative to the preceding parameter candidate, or alternatively, until a specified maximum iteration.

add theoretical Bayes theorem computations here

7.5 Integrating Other Islamic Literatures

The third objective of this paper asks on how to unify all of these results from Statistics, Machine Learning, AI, and the work of Islamic scholars on Qur'ānic studies help in understanding the Qur'ān. Discussions on this will be provided in the next chapter, but in terms of the methodology on how to combine all of this into something that is accessible to Muslims and those interested in learning the Qur'ān, this paper will provide the architecture on how to package all of these.

The idea of combining the results in this paper with the work of Islamic scholars requires some form of relation, that is, relating particular results to what was studied by the scholars before in order to provide more context. For such tasks, of relating particular result to a large corpus of documents written in Qur'ānic studies will seem to be a great endeavor. Manually reading through those corpus and combining to the results of this paper, plus trying to summarize all of this combinations can be a daunting task. Fortunately, at the age of Generative AI, such task can be automated. To do this, there are three main components, and these are:

1. Digitized Pre-Modern Islamic Documents - this include like the *ahadith* or the traditions or *sunna* سُنَّة of the Prophet, and writings of the early Islamic scholars.

2. Information Retrieval Algorithm - an algorithm that will help extract the necessary pre-modern Islamic document related to the results.

3. Text Summarizer - a generative AI model that can create coherent summary on the combinations of the results of this paper plus the context from the pre-modern Islamic document

7.5.1 Open Islamic Text Initiative

The OpenITI or Open Islamic Text Initiative project by Aga Khan University, Institute for the Study of Muslim Civilisations in London, Roshan Institute for Persian Studies at the University of Maryland, College Park, and Universität Hamburg is a huge step to computational analysis of the Islamicate texts, which include the pre-modern Islamic texts. What the team of this project did is that they converted all of the said documents into a text file, so that, it can be used for any digital humanities study of these texts.

According to their About page in their website[4], since its founding in 2016, OpenITI's work has focused on the tasks necessary to build digital capacity in Islamicate studies, including improving Arabic-script optical character recognition (OCR) and handwritten text recognition (HTR), developing robust Arabic-script standards for OCR and HTR output and text encoding, and creating platforms for the collaborative creation of Islamicate text corpora and digital editions.

More specifically, OpenITI compiled all of these digitized texts in their sub-project called KITAB or Knowledge, Information Technology, and the Arabic Book. For this paper, the digitized Sahih al-Bukhari from OpenITI's KITAB will be used as the Islamic context for demonstrating how to combine the results from this study. Fortunately, a Julia library called Kitab.jl[5] developed by Asaad (2022a) was made to interface with KITAB books for easy access, and will be used for this project.

[4] `https://openiti.org/`
[5] `https://github.com/alstat/Kitab.jl`

7.5.2 Retrieval-Augmented Generation

The popularity of Generative AI like ChatGPT has made the said tool the go-to assistant when it comes to anything writing. It is especially good at synthesizing different texts and summarizing it beautifully. However, like any *mathematical models*, it only has knowledge about the data it was trained on. In fact, those that it even learned has the chance of getting it wrong. This is just the nature of models in general, they are not perfect but good enough to be an effective assistant. Therefore, when it comes to new informations, or data, these AI models can still generate answers that seem believable when in fact it is not, and this is what is called as *AI hallucination*. So the question now is, how can we at least mitigate this limitation and still be able to take advantage its creative writing skills? Well, the simplest way is to provide it more context. Like instead of just asking a question to it, provide it with a document as an attached context for it to read, so that, it can provide its answer not only based on the knowledge it learned during the training, but also the documents provided to it in the query. This will make the AI generate a better and more relevant answer.

Now, the next question is, if there are several documents, how can someone automate the process where the user will only query, and then the AI will be given the necessary document it needs for the context, plus those documents that are related to the query can also be presented as part of the outputs? Well, this is the concept of Retrieval-Augmented Generation or RAG. This method proposed by Lewis et al. (2020) is gaining popularity especially in companies trying to integrate AI into their workflow.

add figure here in the context of the methodology proposed

7.6 Programming Languages Setup

This section will discuss the programming languages used in this paper. As mentioned in Chapter 1, there are three main programming languages that will be used, all of which are known in the area of data analysis, and these are Julia, Python, and

Table 7.2: Download page for programming languages used

Software	Version	Download Page
Julia	1.10.3	`https://julialang.org/downloads/`
Python	3.12.2	`https://www.python.org/downloads/`
R	3.4.4	`https://cran.r-project.org/bin/windows/base/`

R. The rule of thumb here is to use Julia as much as possible. With that said, Python is the popular framework for deep learning modeling, and therefore will be used in this paper for such tasks if in case the deep learning model is not available in Julia. Finally, R is known for almost all statistical models, especially those that are very niche, and this is because R was created as a statistical software not as a general purpose programming language like Julia and Python. To download these software, refer to Table 7.2 for the links to the download page. Further, there are several tutorials on these programming languages in Youtube[6] for those interested to learn these. Finally, all of the codes for this paper are stored in a Github repository below:

`https://github.com/alstat/ma-thesis/tree/main/codes`

Further, in terms of the libraries used, QuranTree.jl[7] by Asaad (2021) will be used as the main library for interfacing with the digitized Qur'ān text; Yunir.jl[8] by Asaad (2022b) will be used for text analytics; Kitab.jl[9] by Asaad (2022a) will be used for interfacing with other Islamicate texts from OpenITI[10].

[6] `https://www.youtube.com/`
[7] `https://github.com/alstat/QuranTree.jl`
[8] `https://github.com/alstat/Yunir.jl`
[9] `https://github.com/alstat/Kitab.jl`
[10] `https://openiti.org/`

Chapter 8

Results and Discussions

This chapter is organized as follows: Section 8.1 will discuss the results of the descriptive statistics; Section 8.2 will discuss the results on the morphological analyses; Section 8.3 will discuss the structural analyses including the Mathematical structures of the Qur'ān; Section 8.4 will discuss the results for thematic analyses using both statistical and Large Language models; and finally, Section 8.5 will discuss the use of other Islamic texts that will help in adding more contexts for the Large Language models.

8.1 Descriptive Statistics

This section will focus on the results of the descriptive statistics of the Qur'ān's *ayāt* اَيَات (verses) and *kalimat* كَلِمَات or words. Figure 8.1 visualizes the frequency of the *ayāt* اَيَات and words of the Qur'ān using a combinations of bar, density, and box plots. The figure is divided into three main parts. The first part is the statistics of the *ayāt*'s اَيَات count. It can be seen that in terms of the number of verses, it is generally decreasing just like what the Muslims observed. Table 8.1 summarizes the necessary statistics of Figure 8.1. From the said table, there are 39 *ayāt* اَيَات to expect based on the median statistics, and there are about 55 *ayāt* اَيَات to expect per *sūrah* سُورَة based on the mean statistics. The reason the mean is higher than the median follows from the fact that there are surahs that can be considered outlier because of the large number of *ayāt* اَيَات. The annotation on the *sūrah* سُورَة with the highest number of *ayāt* اَيَات are indicated in Figure 8.1. These *sūrah* سُورَة pushes the mean to higher number than the median. Indeed, these *sūrahs* سُور have stretched the shape of the density and the box plots to higher values, suggesting that the data points are more varied. The width of the density and the box plots is measured by the variance and the standard deviation, which is simply the square root of the variance.

Table 8.1: Descriptive statistics of the *ayāt* اَيَات counts and the counts of its words

Count Data	Mean	Median	Std. Deviation
Ayahs	54.70	39	53.21
Words	679.20	344	931.18
Words per Ayah	10.27	8.23	6.35

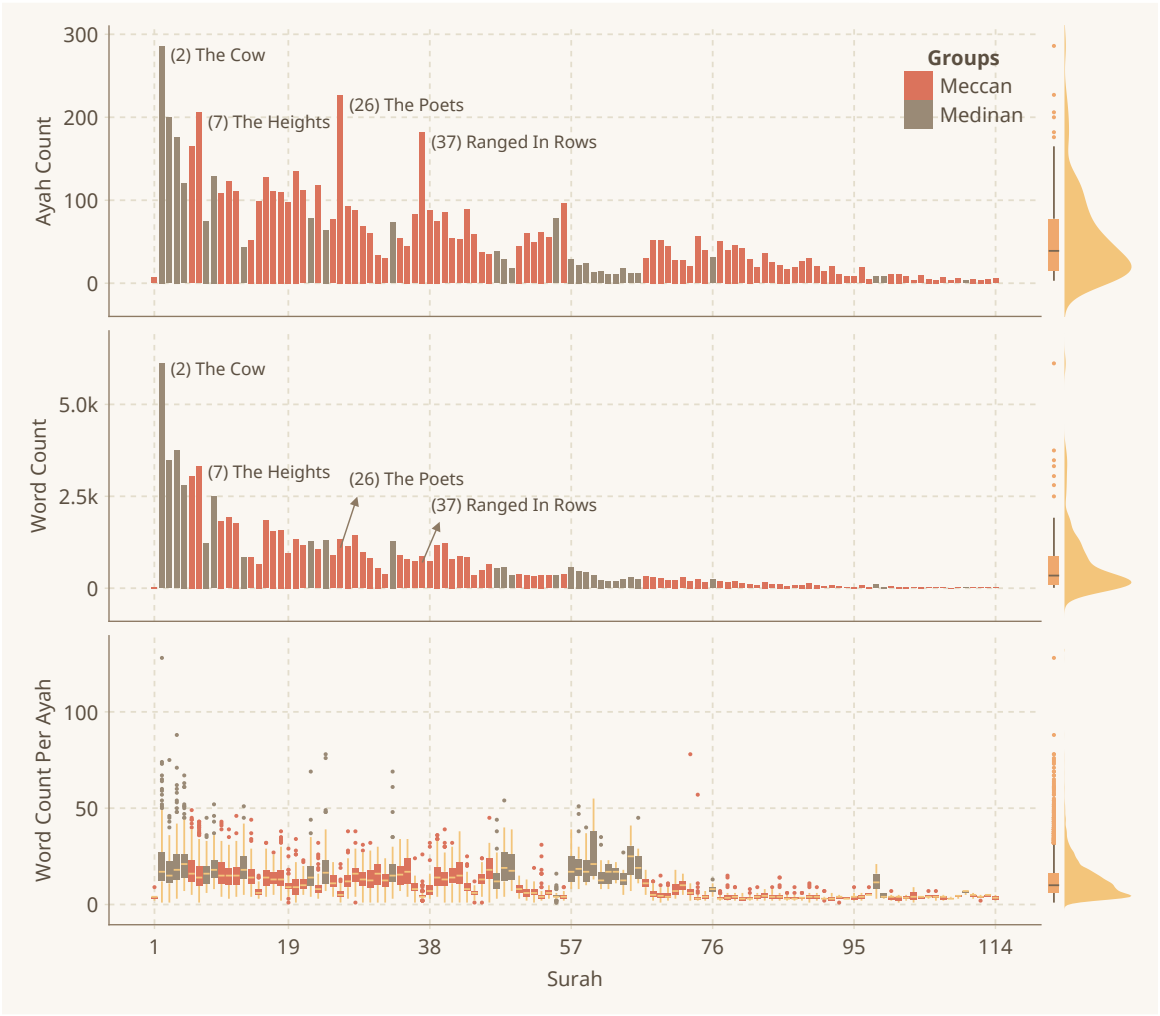


Figure 8.1: Statistics of the words and *ayāt* اَيَات (verses) of the Qur'ān

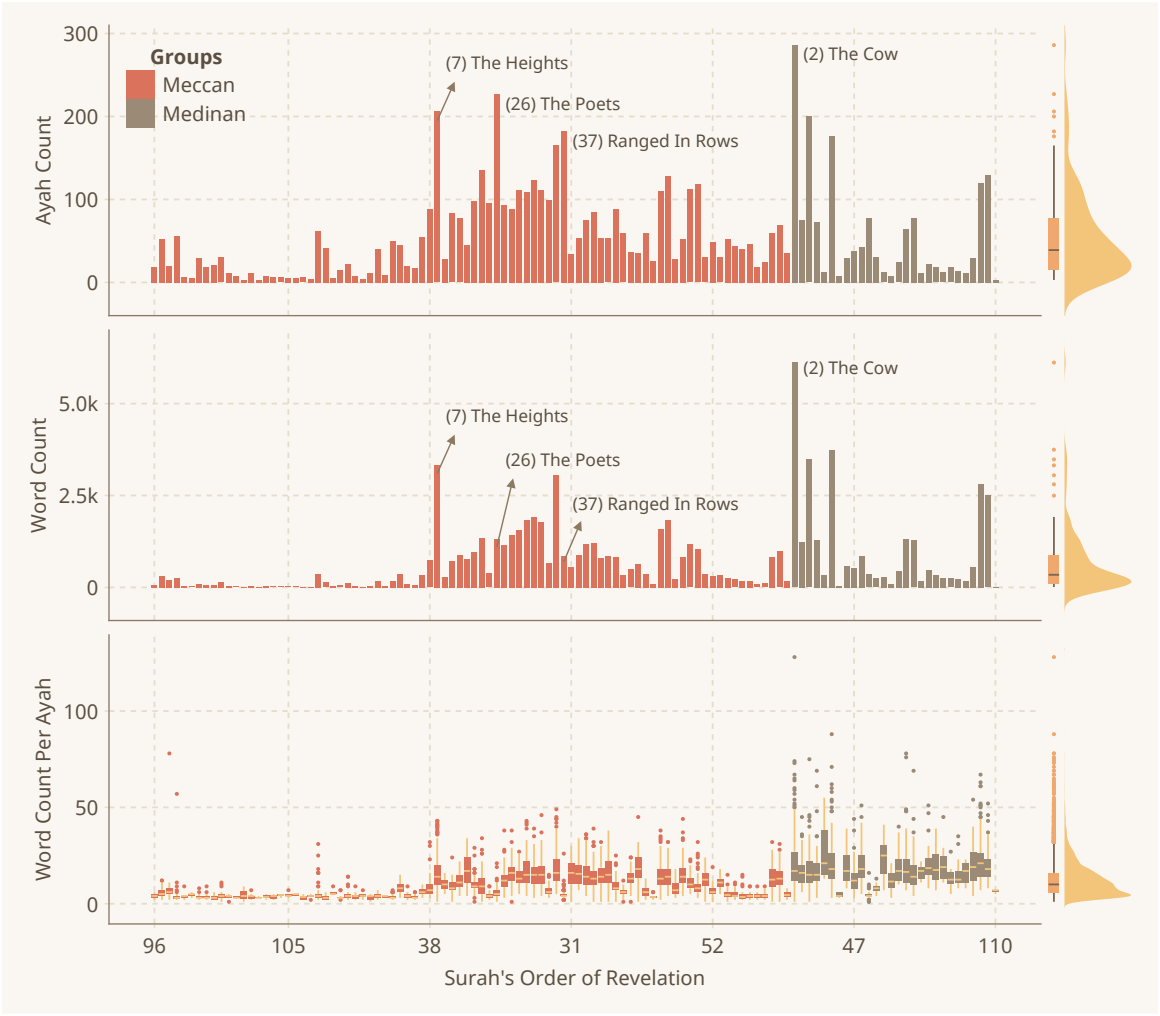


Figure 8.2: Statistics of the words and *ayāt* اَيَات (verses) of the Qur'ān according to revelation order

8.1.1 Verses

8.2 Morphological Analysis

8.3 Structural Analysis

8.3.1 Concentric Structure

8.3.2 Mathematical Structure

8.3.3 Discussions on Islamic Philosophy of Qur'ān's Structural Analysis

Chapter 9

Conclusion and Recommendation

References

Abro, B., Naqvi, A. B., & Hussain, A. (2012). Qur'an recognition for the purpose of memorisation using speech recognition technique. In *2012 15th international multitopic conference (inmic)* (p. 30-34). doi: 10.1109/INMIC.2012.6511440

Afzal, H., & Mukhtar, T. (2019). Semantically enhanced concept search of the holy quran: Qur'anic english wordnet. *Arabian Journal for Science and Engineering*, *44*, 3953–3966.

Ahmed, A. H., & Abdo, S. M. (2017). Verification system for quran recitation recordings. *International Journal of Computer Applications*, *163*(4), 6–11.

Alshammeri, M., Atwell, E., & ammar Alsalka, M. (2021). Detecting semantic-based similarity between verses of the quran with doc2vec. *Procedia Computer Science*, *189*, 351-358. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050921012291 (AI in Computational Linguistics) doi: https://doi.org/10.1016/j.procs.2021.05.104

Asaad, A.-A. B. (2021). QuranTree.jl: A Julia package for Quranic Arabic corpus. In *Proceedings of the sixth arabic natural language processing workshop* (pp. 208–212). Kyiv, Ukraine (Virtual): Association for Computational Linguistics. Retrieved from https://aclanthology.org/2021.wanlp-1.22

Asaad, A.-A. B. (2022a, June). *Kitab.jl: A Julia interface to the Open Islamic Texts Initiative (OpenITI) data.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.6665710 doi: 10.5281/zenodo.6665710

Asaad, A.-A. B. (2022b, June). *Yunir.jl: A lightweight Arabic NLP toolkit for Julia.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.6629868 doi: 10.5281/zenodo.6629868

Bashir, M. H., Azmi, A. M., Nawaz, H., Zaghouani, W., Diab, M., Al-Fuqaha, A., & Qadir, J. (2023). Arabic natural language processing for qur'anic research: a systematic review. *Artificial Intelligence Review*, *56*(7), 6801–6854.

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions*, *53*, 370-418. Retrieved from http://www.jstor

.org/stable/105741

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. Retrieved from https://epubs.siam.org/doi/10.1137/141000671 doi: 10.1137/141000671

Birmingham University. (2015). *Birmingham qur'an manuscript dated among the oldest in the world.* Birmingham University. (Available at: https://www.birmingham.ac.uk/news-archive/2015/birmingham-quran-manuscript-dated-among-the-oldest-in-the-world (Accessed: July 8th, 2023))

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Cooperman, A., O'Connell, E., & Stencel, S. (2011). *the future of the global muslim population* (Tech. Rep.). Pew Research Center.

Darwish, K., Habash, N., Abbas, M., Al-Khalifa, H., Al-Natsheh, H. T., Bouamor, H., ... Mubarak, H. (2021, mar). A panoramic survey of natural language processing in the arab world. *Commun. ACM*, *64*(4), 72–81. Retrieved from https://doi.org/10.1145/3447735 doi: 10.1145/3447735

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dukes, K., & Habash, N. (2010, May). Morphological annotation of Quranic Arabic. In N. Calzolari et al. (Eds.), *Proceedings of the seventh international conference on language resources and evaluation (LREC'10)*. Valletta, Malta: European Language Resources Association (ELRA). Retrieved from http://www.lrec-conf.org/proceedings/lrec2010/pdf/276_Paper.pdf

ELAffendi, M. A., Abuhaimed, I., & AlRajhi, K. (2021). A simple galois power-of-two real time embedding scheme for performing arabic morphology deep learning tasks. *Egyptian Informatics Journal*, *22*(1), 35-43. Retrieved from https://www.sciencedirect.com/science/article/

pii/S1110866520301146 doi: https://doi.org/10.1016/j.eij.2020.03.002

Farrin, R. (2014). *Structure and qur'anic interpretation: A study of symmetry and coherence in islam's holy text*. White Cloud Press. Retrieved from https://books.google.com.ph/books?id=io03ngEACAAJ

Haykin, S. (1998). *Neural networks: A comprehensive foundation* (2nd ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.

Karatay, F. E. (1962). *Topkapı sarayı müzesi kütüphanesi arapça yazmalar katalogu. kur'an, kur'an ilimleri, tefsirler no. 1 - 2171*. Küçükaydın Matbaası, İstanbul.

Khalifa, R. (1981). *The computer speaks: God's message to the world*. Renaissance Production.

Laplace, P. S. (1986, 08). Memoir on the probability of the causes of events. *Statist. Sci.*, *1*(3), 364–378. Retrieved from http://dx.doi.org/10.1214/ss/1177013621 doi: 10.1214/ss/1177013621

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 9459–9474). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

Manila Bulletin. (2022). *PH embassy in riyadh hosts first asian qur'an memorization contest.* Manila Bulletin. (Available at: https://mb.com.ph/2022/04/30/ph-embassy-in-riyadh-hosts-first-asian-quran-memorization-contest/ (Accessed: July 8th, 2023))

Mannaa, Z. M., Azmi, A. M., & Aboalsamh, H. A. (2022). Computer-assisted i'raab of arabic sentences for teaching grammar to students. *Journal of King Saud University - Computer and Information Sciences*, *34*(10, Part B), 8909-8926. Retrieved from https://www.sciencedirect.com/science/article/pii/S1319157822002944 doi: https://doi.org/10.1016/j.jksuci.2022.08.020

McCulloch, W. S., & Pitts, W. (1943, December 21). A logical calculus of the ideas

immanent in nervous activity. *Bulletin of Mathematical Biology*, *5*(4), 115–133. doi: 10.1007/bf02478259

Muirhead, R. J. (2005). *Aspects of multivariate statistical theory*. Hoboken, New Jersey: John Wiley & Sons, Inc.

Nassourou, M. (2011). *Using machine learning algorithms for categorizing quranic chaptersby major phases of prophet mohammad's messengership.*

Neuwirth, A. (2007). *Studien zur komposition der mekkanischen suren*. Berlin, Boston: De Gruyter. Retrieved 2023-07-09, from https://doi.org/10.1515/9783110920383 doi: doi:10.1515/9783110920383

Nielsen, M. A. (2015). *Neural network and deep learning*. Determination Press. Retrieved from http://neuralnetworksanddeeplearning.com/

R Core Team. (2023). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from https://www.R-project.org/

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Sadeghi, B., & Bergmann, U. (2010). The codex of a companion of the prophet and the qurān of the prophet. *Arabica*, *57*(4), 343 - 436. doi: \url{https://doi.org/10.1163/157005810X504518}

Shahzadi, N., ur Rahman, A., & Sawar, M. J. (2012). Semantic network based classifier of holy quran. *International Journal of Computer Applications*, *39*, 43-47. Retrieved from https://api.semanticscholar.org/CorpusID:6539715

Sharaf, A., & Atwell, E. (2009). Knowledge representation of the quran through frame semantics: a corpus-based approach. In *Proceedings of the fifth corpus linguistics conference.* The Fifth Corpus Linguistics Conference. Retrieved from https://api.semanticscholar.org/CorpusID:18278736

Sharaf, A., & Atwell, E. (2012a, May). QurAna: Corpus of the Quran annotated with pronominal anaphora. In N. Calzolari et al. (Eds.), *Proceedings of the eighth international conference on language resources and evaluation* (*LREC'12*)

(pp. 130–137). Istanbul, Turkey: European Language Resources Association (ELRA). Retrieved from http://www.lrec-conf.org/proceedings/lrec2012/pdf/123_Paper.pdf

Sharaf, A., & Atwell, E. (2012b, May). QurSim: A corpus for evaluation of relatedness in short texts. In N. Calzolari et al. (Eds.), *Proceedings of the eighth international conference on language resources and evaluation* (*LREC'12*) (pp. 2295–2302). Istanbul, Turkey: European Language Resources Association (ELRA). Retrieved from http://www.lrec-conf.org/proceedings/lrec2012/pdf/190_Paper.pdf

Siddiqui, M. A., Faraz, S. M., & Sattar, S. A. (2013). Discovering the thematic structure of the quran using probabilistic topic model. In *2013 taibah university international conference on advances in information technology for the holy quran and its sciences* (p. 234-239). doi: 10.1109/NOORIC.2013.55

Sidky, H. (2020). On the regionality of qur'anic codices. *Journal of International Quranic Studies Association*. doi: http://dx.doi.org/10.5913/jiqsa.5.2020.a005

Sinai, N. (2014). When did the consonantal skeleton of the quran reach closure? part ii. *Bulletin of the School of Oriental and African Studies*, *77*(3), 509–521. doi: 10.1017/S0041977X14000111

Sinai, N. (2017). *The qur'an: A historical-critical introduction*. Edinburgh University Press Ltd.

Thabet, N. (2004). Stemming the qur'an. In *Proceedings of the workshop on computational approaches to arabic script-based languages* (p. 85-88). USA: Association for Computational Linguistics.

Thabet, N. (2005). Understanding the thematic structure of the qur'an: an exploratory multivariate approach. In *Proceedings of the acl student research workshop* (p. 7-12). USA: Association for Computational Linguistics.

Van Rossum, G., & Drake Jr, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polo-

sukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wansbrough, J. (2004). *Quranic studies: Sources and methods of scriptural interpretation*. Prometheus Books.

Wikipedia. (2015). *Comparison of a 20th-century edition of the quran (left) and the birmingham quran manuscript (right)*. Wikipedia. (Available at: `https://en.wikipedia.org/wiki/Birmingham_Quran_manuscript` (Accessed: July 9th, 2023))