NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

A Study on chromatin structure via nucleosome positioning pattern classification using n-gram graphs

Nikolaos-Stefanos M. Kostagiolas

Supervisors: **Stamatopoulos Panagiotis**, Assistant Professor, UOA
Giannakopoulos George, PhD., NCSR Demokritos

ATHENS

MAY 2017

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μελέτη της δομής της χρωματίνης μέσω αναζήτησης μοτίβων νουκλεοσωμάτων με τη χρήση γράφων ν-γραμμάτων

Νικόλαος-Στέφανος Μ. Κωσταγιόλας

Επιβλέποντες: **Σταματόπουλος Παναγιώτης,** Επίκουρος Καθηγητής, ΕΚΠΑ
Γιαννακόπουλος Γεώργιος, Δρ., ΕΚΕΦΕ Δημόκριτος

ΑΘΗΝΑ
ΜΑΪΟΣ 2017

BSc THESIS

A Study on chromatin structure via nucleosome positioning pattern classification using n-gram graphs

Nikolaos-Stefanos M. Kostagiolas

S.N.: 1115201100039

SUPERVISORS:

Stamatopoulos Panagiotis , Assistant Professor, UOA
Giannakopoulos George , PhD., NCSR Demokritos

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μελέτη της δομής της χρωματίνης μέσω αναζήτησης μοτίβων νουκλεοσωμάτων με τη χρήση γράφων ν-γραμμάτων

Νικόλαος-Στέφανος Μ. Κωσταγιόλας
Α.Μ.: 1115201100039

ΕΠΙΒΛΕΠΟΝΤΕΣ:

Σταματόπουλος Παναγιώτης, Επίκουρος Καθηγητής, ΕΚΠΑ
Γιαννακόπουλος Γιώργος, Διδάκτωρ, ΕΚΕΦΕ Δημόκριτος

ABSTRACT

Knowing the exact locations of nucleosomes in a genome is crucial for understanding how gene expression is organized. Consequently, during the latest years, an increasing interest has emerged, in applying text-mining techniques in genomic studies, an example of which is examining whether the primary structure of DNA, i.e. textual data extracted from genomes, influences nucleosome positioning and, thus, chromatin structure. To the best of our knowledge, there exists no complete study on the effect of representation to the classification of genomic sequenses as nucleosome-free regions (NFR) - i.e. sequences depleted of nucleosomes - or nucleosome-binding sites (NBS) - i.e. sequences where nucleosomes are present. In this thesis we study 3 different genomic sequence representations (Hidden Markov Models, Bag-of-Words and N-gram Graphs) in combination to a number of machine learning algorithms on the task of classifying genomic sequences as NFR and NBS. Finally, we conclude that, based on our findings, different approaches that involve the usage of different representations or algorithms can be more or less effective at predicting nucleosome positioning based on the textual data of the underlying genomic sequence.

SUBJECT AREA: Machine Learning, Molecular Biology
KEYWORDS: e.g. nucleosomes, genomic sequences, classification, representations, machine learning

ΠΕΡΙΛΗΨΗ

Γνωρίζοντας τις ακριβείς θέσεις των νουκλεοσωμάτων σε ένα γονιδίωμα είναι το κλειδί για την κατανόηση του πώς τα γονίδια ρυθμίζονται. Κατά συνέπεια, κατά τη διάρκεια των τελευταίων ετών, ένα αυξανόμενο ενδιαφέρον έχει προκύψει, για τις εφαρμογές των τεχνικών εξόρυξης κειμένου στις γονιδιωματικές μελέτες, ένα παράδειγμα των οποίων είναι η εξέταση του αν η πρωτοταγής δομή του DNA, δηλαδή τα δεδομένα κειμένου που προκύπτουν από γονιδιώματα, επηρεάζει τις θέσεις των νουκλεοσωμάτων και συνεπώς τη δομή της χρωματίνης. Βάσει των γνώσεών μας, δεν υπάρχει πλήρης μελέτη που να εξετάζει τα αποτελέσματα διαφορετικών αναπαραστάσεων στην ταξινόμηση γονιδιωματικών ακολουθιών ως Περιοχών Ελεύθερες Από Νουκλεοσώματα (ΠΕΑΝ) ή Περιοχών Πρόσδεσης Νουκλεοσωμάτων (ΠΠΝ). Το θέμα της Πτυχιακής Εργασίας αυτής, είναι η μελέτη 3 διαφορετικών αναπαραστάσεων γονιδιωματικών ακολουθιών σε συνδυασμό με έναν αριθμό αλγόριθμων μηχανικής μάθησης στο πρόβλημα της ταξινόμησης γονιδιωματικών ακολουθιών ως ΠΔΝ ή ΠΠΝ. Τέλος, καταλήγουμε στο συμπέρασμα ότι, με βάση τα ευρήματά μας, διαφορετικές προσεγγίσεις με χρήση διαφορετικών αναπαραστάσεων ή αλγορίθμων μπορούν να είναι λιγότερο ή περισσότερο αποτελεσματικές στην πρόβλεψη των θέσεων που καταλαμβάνουν τα νουκλεοσώματα, βάσει των δεδομένων κειμένου της υποκείμενης γονιδιωματικής αλληλουχίας.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μηχανική Μάθηση, Μοριακή Βιολογία

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: νουκλεοσώματα, γονιδιωματικές ακολουθίες, ταξινόμηση, αναπαραστάσεις, μηχανική μάθηση

Στους γονείς μου Μιχάλη και Λίνα, στον αδελφό μου Διονύση και στην αγαπημένη μου Εύα, χωρίς τη συμπαράσταση των οποίων η πτυχιακή αυτή δε θα μπορούσε να ολοκληρωθεί.

ACKNOWLEDGEMENTS

For the completion of this B.Sc. Thesis I first and foremost want to thank my supervisor at the NCSR Demokritos, Dr. George Giannakopoulos, thanks to the guidance of whom I managed to withstand all the tests that surfaced in this project. Armed with insight, concise in his words and especially patient and loyal towards myself, not only he managed to make me love the research procedure but also helped me acquire feats which I will surely carry with me forever, as his former student, as a new and aspiring scientist and, mainly, as a human being.

Special acknowledgements also have to go towards Dr. Christophoros Nikolaou, Assistant Professor of Molecular, Cellular and Developmental Biology and Biochemistry of the Biology Department at the University of Crete, on the findings of whom this B.Sc. thesis was based. Professor Nikolaou not only made the essential experimental data available to me but he also offered me useful advice during the time I was studying the basic elements of Molecular Biology.

However, I also have to greatly thank my supervisor at the Department of Informatics and Telecommunications , Assistant Professor Panagiotis Stamatopoulos, who, at the beginning of my search for a potential thesis subject, he guided me towards coming in touch and meeting Dr. Giannakopoulos. I also have to thank him for the continuous communication and conversations we had throughout the duration of my thesis, especially in matters involving the Machine Learning field.

Last but not least, I am obliged towards Ilias Roussos, secondary education Professor of Chemistry and Biology, for the essential information he provided me with, at the very beginning of my quest, and for helping me get a first glimpse of the field of Molecular Biology.

There are a few occasions in which a human can possibly thank his beloved ones and it may be out of context to do it here. Nevertheless, I have to thank all the people who are mentioned in my dedication for the undivided support, tolerance and patience they showed during this year's quest towards completing this thesis. Their contribution is multiplied, and this is due to their effort in guiding me towards the realization that, in order to support a human being in anything, any knowledge about the path leading to the solution of his/her problem is not necessary at all. The thing that is most important though is to be on his side, despite being ignorant. And this is the main reason I have to thank these people for.

CONTENTS

1 Introduction **19**

2 Related Work **23**

3 Proposed Approach **29**

3.1 Method Description 29

3.2 Representations and Features 30

3.2.1 Vector Space Models 30

3.3 Hidden Markov Models - HMMs 31

3.4 Bag-of-words 37

3.5 N-gram Graphs 41

3.6 Classification Algorithms 46

3.6.1 Decision Trees 46

3.6.2 Naive Bayes 47

3.6.3 k-Nearest Neighbor 48

3.6.4 Support Vector Machines 49

3.7 Evaluation 51

3.7.1 Evaluation Metrics Definition 51

3.7.2 What do different metrics mean ? 52

4 Experiments **55**

4.1 Experimental Setup 55

4.2 Results 56

5 Discussion and Conclusion **69**

ABBREVIATIONS - ACRONYMS **71**

BIBLIOGRAPHY **73**

LIST OF FIGURES

1.1 **The crystal structure of the nucleosome core particle** (Source: Wikipedia) 19

3.1 **Vector representation of three-dimensional entity space** 31
3.2 **The above example.** 33
3.3 **The Hidden Markov Model in our approach** 35
3.4 **Graph extracted from the genomic sequence ATTCGC based on the symmetric type of window.** 43
3.5 **A decision tree** 46
3.6 **Training set** 47
3.7 **Confusion matrix** 51
3.8 **Confusion matrix of model A** 52
3.9 **Confusion matrix of model B** 52
3.10 **Confusion matrix of model C** 52

4.1 **Recall measure per Representation (values between 0.00 - 0.65 neglected due to lack of data).** 57
4.2 **Precision measure per Representation.** 59
4.3 **F-measure per Representation (NOTE: minimum y-axis value is 0.70).** 61
4.4 **Training Time per Algorithm/Representation Pair.** 63
4.5 **Classification Time per Algorithm/Representation Pair.** 64
4.6 **Total Time per Algorithm/Representation Pair.** 66

LIST OF TABLES

3.1 ***Notation used in the HMM chapter.*** 34

3.2 **N-gram examples from various disciplines.** 42

4.1 **Mean Recall per Algorithm/Representation Pair (higher is better).** . . 58

4.2 **Mean Precision per Algorithm/Representation Pair (higher is better).** 60

4.3 **Mean F-measure per Algorithm/Representation Pair (higher is better).** 62

4.4 **Mean Classification Time per Algorithm/Representation Pair (lower is better).** 65

4.5 **Mean Total Time per Algorithm/Representation Pair (lower is better).** 67

1. INTRODUCTION

It has been suggested recently [1] [2], that nucleosome positioning in the eukaryotic genome playes a vital role in fundamental processes at the molecular level, such as transcriptional regulation or replication.

A **Nucleosome** is a proteinic structure which plays a vital role in DNA compression in eukaryotic nuclei. The DNA sequence encircles these units resembling a thread wrapped around a spool.

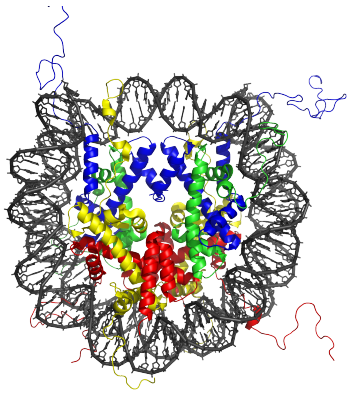

Figure 1.1: The crystal structure of the nucleosome core particle (Source: Wikipedia)

Nucleosomes constitute the basic structural units of chromatin at the eukaryotic genome, a molecular complex which is used to envelop the genomic code into the nucleus while still keeping it appropriately accessible. When folded in a sequence of overlapping higher order structures, nucleosome layers ultimately create chromosomes; this is responsible for both ensuring DNA density and the creation of an added layer of regulatory control, which ensures the proper expression of genes.

DNA, Histones and Chromatin

Histones are proteins which compact chromosomal DNA into the microscopic space of the eukaryotic nucleus. The combination of these proteinic structures and DNA forms a complex which is called **chromatin**.

Processes such as **transcription** and **replication** allow DNA to be copied by allowing chromatin to unfold. During this process the presence of **nucleosomes** pose barriers to determining which parts of DNA are to be copied. The parts of DNA which are to be copied during transcription are called **transcription sites**.

What happens after DNA Regulation/Replication ?

The genetic information copied after DNA regulation or replication is used in **gene expression**, i.e. the process which controls the synthesis of a functional gene product, usually a protein. Genetic information can be transmitted from one generation to the next without being altered, influencing the attributes of the offspring. This shared information between the two generations is called **epigenetically inherited information**.

Frequently, the particular positioning of nucleosomes has been shown to be driven by the properties of the underlying sequence, but it remains under question as whether DNA has a decisive effect on such positioning.

In our work, we're aiming to reveal the features in the DNA sequences that guide nucleosomes to be formed or not. To achieve this feat we distinguish genomic data into two classes : **Nucleosome Free Regions** or simply NFR and **Nucleosome Binding Sites** or simply NBS in order to examine, with the help of various Machine Learning algorithms, the extent to which the primary DNA structure contains patterns that appear to guide nucleosome positioning in the eukaryotic genome.

Below we elaborate in what way our work complements all the research efforts made in nucleosome positioning studies during a period spanning from 2002 to 2014, and how it brings novelty with respect to the approaches taken so far

A notable factor that helped the field accelerate its findings is the rise in efficiency and accuracy of computational methods in analyzing and predicting nucleosome positioning. Fast software tools have been developed in order to facilitate even more complex experiments which opt to tackle more obscure questions by utilizing more and more detailed and data-heavy genome maps. However, it would be interesting to examine, to what extent different Machine Learning algorithms and representations would be able to predict nucleosome positioning based on the textual data of the underlying DNA sequence.

As one can note, the research made so far has resorted to the established methods of Hidden Markov Models (HMMs) and Support Vector Machines [3] for classification. In our study we will use three different kinds of representations :

- Hidden Markov Models (HMMs),
- N-gram Graphs (NGGs) and,
- Bag-of-Words or Bag-of-Nucleotides (BOWs)

along with four different kinds of classification algorithms :

- Decision Trees,
- Naive Bayes,
- k-Nearest Neighbors and,
- Support Vector Machines

We selected different algorithms from different approaches on machine learning and representations supporting varying use of contextual information. This was done to evaluate the pros and cons of combinations through a unified set of experiments and evaluations. This can also give us many hints about nucleosome positioning motifs that are still obscure and may lead to future research insights that could shed light in genomic studies.

Here we're going to discuss the structure of the rest of the thesis. We begin by focusing on related work and how this thesis compliments it, in chapter 2. Then, we proceed by presenting the methods we used in our approach, in chapter 3. Finally, we provide the results of our experiments including a discussion and conclusion based on them, in chapters 4 and 5 respectively.

2. RELATED WORK

Knowing the exact locations of nucleosomes in a genome is crucial for understanding how gene expression is organized. Recent discoveries have led to understanding that chromatin structure is regulated by some basic principles. In this chapter we will discuss about certain methods which facilitate us to produce more and more accurate results and build an intuition surrounding nucleosome positioning and gene regulation, while trying to give a brief historical resume of the research made in this field.

Althought it was argued early on that nucleosomes are positioned along DNA randomly, it soon became apparent that chromatin structure is indeed regulated by some certain principles which are also responsible for essential biological reactions such as transcription, recombination and replication. In recent studies, it has been suggested that chromatin structure plays a vital role in fundamental molecular processes, such as transcriptional regulation [1] or replication [2].

In the first case, *Guenther et al.* [1] provided the results of a genome-wide analysis of *H.sapiens* cells, which indicated that nucleosomes flank the transcription initiation sites (i.e. sites where transcription begins), where transcription is more likely to occur, as these parts of the DNA are more accessible to transcription factors. This provided strong evidence that regulation happens in certain sites, which could possibly be defined by nucleosome positioning, and not in the whole genome.

Furthermore, according to the study made by *Eaton et al.* [2], which attempted to analyze the molecular process of replication in *S.cerevisiae*, it was found that nucleosomes also tend to flank the replication origins (i.e. sites where replication begins) in the genome. In addition to that, the genomic sequences present at the origin sites were depleted of nucleosomes, indicating that a nucleosome-free region was present.

Important sites in the Genome

Some important sites in genome maps are **transcription initiation sites** (or **transcription start sites**) and **replication origins**, which flag where the transcription and replication processes begin in the DNA sequence, respectively. Adjacent to these

sites are situated the **promoters**, regions of DNA which are responsible for initiating the transcription of a particular gene.

The proteins in a genome that are essential for the regulation of the expression of genes near a sequence, by binding to it, are called **transcription factors**.

Although these studies indicated that the functions of regulation and replication may be defined by nucleosome organization, the principles that specify the location of nucleosomes along the DNA sequence remain obscure.

In an initial study made by *Vicent et. al* [4] on Mouse Mammary Tumor Virus (MMTV) promoter shed some light to the factors that specify nucleosome positioning, by suggesting that the position of nucleosomes is driven to an extent by the underlying DNA sequence principles. However, to what extent nucleosomal DNA accessibility is influenced by the nucleotide sequence of the DNA remains an open question.

Upon 2005, it became apparent that genome-wide nucleosomal maps may play a vital role in providing us with information regarding the relationship between chromatin structure, histone modifications and the control of gene expresion. Large-scale studies of nucleosome positions featuring such maps succeeded in defining the positions occupied by nucleosomes for a significant portion of the *S.cerevisiae* genome [5], while others provided higher resolution maps describing nucleosome positioning along the entire yeast genome [6]. Finally, in [7] *Shivaswamy et al.* used high-throughput sequencing to map the remodeling of nucleosomes (i.e. the positions taken by the nucleosomes before and after a change) after genome-wide transcriptional changes had occured.

In the first study, *Yuan et al.* [5] denoted that transcription factor binding sites were lacking nucleosomes, suggesting that the positions occupied by nucleosomes determine the transcription factor accessibility in a global scale. This was strongly emphasized by the fact that these nucleosome-free sequences were evolutionary conserved.

Furthermore, *Lee et al.*'s [6] study confirmed the above results, by showing that a general pattern of nucleosome occupancy was present at the borders of the transcription factors, with the nucleosomes being fixed at the transcription start sites (or TSS). It was also revealed that this pattern was shared between functionally related genes, at their promoter regions.

Nucleosome positioning patterns in relation to transcribed regions and transcription binding sites were also observed in the study made by *Shivaswamy et al.* [7], in which it was indicated that, during a transcriptional change, nucleosomal rearrangements did not happen across the whole genome, but only in specific sites. However, despite the fact that these sites included transcription factor binding sites, nucleosome remodeling also occurred unexpectedly at promoters even where no apparent transcriptional change had taken place. This provided evidence that the relationship between chromatin remodeling and transcriptional activity could not be driven by globally applicable rules.

While the aforementioned results were more complicated than expected *Stein et al.* [8] used independent nucleosomal datasets of the *S.cerevisiae* genome in an attempt to de-

fine whether nucleosome posititioning was determined by the underlying DNA sequence. The inconsistency in the results between the different datasets used led to the conclusion that DNA sequence preferences have small effect in nucleosome organization *in vivo*, although nucleosome positioning in genomes is not randomly regulated. The paper indicates that more complex experiments on wide data could shed light to the questions concerning the plausibility that DNA preferences organize the position of the nucleosomes across the genome.

In-vivo vs. in-vitro vs. in-silico

Processes which are performed or take place inside an organism can be described by the term **in-vivo**.

When we perform a procedure in an environment which is controlled outside of a living organism, we refer to it as being an **in-vitro** process.

Lastly, when we use computer models or simulations to conduct a scientific experiment, we can describe this method with the term **in-silico**.

At this point, studies began to focus on attempts to define the sequence properties of nucleosomal DNA, mainly by predicting the positions occupied by nucleosomes [9, 10, 11, 12, 3] . Although various computational methods had been utilized, varying from likelihood models [10], [11] to supervised learning strategies [12], [3], the results can be summarized into two basic conclusions:

- we may be able to predict only a subgroup of nucleosome positions
- although there is a possibility for the existence of constraints on the underlying DNA sequence, apparently they are very weak

In the first study, *Caserta et al.* [9] suggested that a specific element, that of the enchancement of dinucleotides which contain AT, was a distinct feature of the sequence code for nucleosome binding sites. That particular signature was a preference associated with nucleosome positioning anchored at the flanks of promoter regions indicating that it can be a possible feature which contributes to the nucleosomal occupancy in such positions.

In addition, *Kaplan et al.*'s [10] study sought to establish the extent to which the DNA sequence determines nucleosome organization in living cells, by comparing results between *in vivo* and *in vitro* datasets of nucleosome positions. While the aforementioned datasets were of different organisms, the *in vivo* belonging to the genome of *C.elegans* and the *in vitro* to the purified genome of *S.cerevisiae*, the results indicated that nucleosomal sequence preferences in both cases were highly correlated. The combined results showed that a sequence preference for approximately 40000 double-stranded 150-base-pair oligonucleotides [1] was apparent, indicating that the underlying DNA sequence has a central role in determining nucleosomal positioning *in vivo*.

[1] a part of DNA whose molecules contain a relatively small number of nucleotides.

Meanwhile, supervised learning began to play its part in the identification of DNA seuquences that either assist or restrain nucleosome formation, as *Peckam et al.* [3] used DNA sequences of *S.cerevisiae* to train a support vector machine, in order to discover the likelihood of nucleosome formations for any given DNA sequence. The results, despite successfully predicting the experimentally determined nucleosome positions accross a specific promoter region, reinforced the fact suggested by previous studies, that nucleosome positioning rules, although existent, may affect just a small percentage of the nucleosome positions in the genome. Another suggestion that became apparent by examining the results was that A/T nucleotides seem to favor nucleosome formation, while G/C ones appear to inhibit it.

In a subsequent study, *Ogawa et al.* [12] followed by refining the method of *Peckam et al.* providing more accurate results achieved with less available data. The results they provided, showed high frequencies of nucleotide G/C, and low frequencies of nucleotide A/T in nucleosomal DNA, relative to the frequencies found in the rest of the genome, which was consistent with the results provided by the previous study. This tendency for G/C containment could possibly indicate a specific sequence preference of nucleosomal DNA.

An alternative approach based on likelihood models was taken by *Segal et al.* [11], attempting to define the sequence preferences responsible for higher nucleosome occupancy. The results of this study demonstrated that 50% of the *in vivo* nucleosome positions are organized by a genomic code, which probably regulates multiple functions including transcription factor binding, transcription initiation and even nucleosome remodeling [2]. Nevertheless, the paper suggests that nucleosome positioning is encoded in eukaryotic genomes, agreeing with all previous studies in that matter.

A more recent study by *Bettecken and Trifonov* [13] which combined various datasets attempted to provide a framework for the sequence elements that drive nucleosome positioning on DNA. However, despite being in line with the findings of previous studies concerning the fact that dinucleotide periodicities play a role in nucleosome organization, no specific preferences for nucleosome binding DNA where accounted. The paper, though, concluded by stating that the selection of the dinucleotides which contribute to the regulation of nucleosome positions is species specific, and it may also differ from region to region, depending on the sequence context.

To sum up, the remarks made by the aformentioned research groups suggest that there is little consistency in nucleosome positioning across the eukaryotic cellular population, indicating that most of the nucleosomes are positioned in a stochastic way and that sequence elements do not provide assistance in defining a wide genomic code that drives the organization of nucleosomal positions.

Indeed, the idea that the overall nucleosome positioning is statistically guided by a subset of nucleosomes has its roots in the early work by *Kornberg and Stryer* [14], while

[2]Nucleosome remodeling is the dynamic modification of chromatin architecture to allow access of condensed genomic DNA to the regulatory transcription factors, and thereby control gene expression (Source:Wikipedia)

further studies [15], elaborated the same model. Furthermore *Reynolds et al.* [16] provided a brief but comprehensive theoretical framework, in order to differentiate between nucleosome-binding sites (NBSs) and nucleosome-free regions (NFRs), and restated the fact that only a small subset of the nucleosomal population was well positioned. However, this study didn't clarify whether any form of constraint was present in the nucleosome positioning patterns. An answer to this question was given in [17], a study which revealed the fact that constraints in the nucleosomal landscape were indeed extant, as its results indicated that consistent nucleosomes may be organized to some extent statistically by nearby nucleosome-free regions.

More specifically, in their 1988 study, *Kornberg and Stryer* [14] proposed some relatively early theoretical models for the statistical positioning of nucleosomes, but due to lack of data, it proved difficult to specify the sequence preferences which guided nucleosome organization and posed stuctural constraints on chromatin patterns. This particular attempt, though, proved to be one of the first to indicate the stochastic nature of nucleosome organization within the genome.

Later, *Mavrich et al.* [15] attempted to establish rules responsible for nucleosomeal organization using a large dataset of immunopurified [3] *S.cerevisiae* nucleosomes. This study suggested that the positions occupied by the -1 and +1 nucleosomes (i.e. the nucleosomes that are situated right before and right after a nucleosome free region) were defined by the underlying genomic sequence and, more importantly, it indicated that the +1 nucleosome imposes statistical rules under which a subset of the nucleosomal population positions are organized statistically, with the constraint decaying as nucleosome positions distance themselves from the barrier formed by the +1 nucleosome.

Furthermore, *Reynolds et al.* [16], in their 2010 paper attempted to provide a more accurate description of a nucleosome positioning pattern, based on the preferences of the underlying sequence, by defining a nucleosomal organization model based on mono-, di- and tri- nucleotide elements and predicting the positions of the nucleosomal population in datasets of *S.cerevisiae* and *H.sapiens*. The results produced achieved higher accuracy than that of two previous nucleosome positioning pattern models combined and suggested that, in both cases, only a small fraction of the nucleosomal population were well-positioned, while the rest of the bulk appeared to be statistically situated across the genome.

Finally, in the study made by *Nikolaou et al.* [17], a comparison of the complete sets of nucleosome positions in the genome of *S.cerevisiae* in two independent experiments produced results which were in unison with the above studies. These results not only included the fact that only a subset of nucleosome positions were consistent cross-experimentally, but also were shown to impose structural and sequential rules. In addition, the presence of consistent nucleosomes in sequences where certain pairs of dinucleotides were also present was once more indicated. Although consistent with previous studies, the most striking discovery of Nikolaou's [17] research team was that a subset of the nucleosome population which was positioned adjacently of nucleosome free regions appeared to be

[3] immunopurification: the use of immunological techniques to purify proteins (Source: Wiktionary.org

highly consistent with regard to the low levels of conservation of the sequence in general. That particular element indicated the presence of particular constraints that may be imposed to consistent nucleosome positioning by nearby nucleosome-free regions through statistical positioning.

In conclusion, it is made clear that, although the existence of a DNA code responsible for organizing nucleosomal positioning is quite plausible, the information load which it could pose on the genetic code by regulating all the positions of the nucleosomal population is a fact that discourages this theory. This indicates that the theory concerning the statistical organization of the bulk, in addition to the sequence-dictated positioning of just a subset of nucleosomes, is more feasible. Nucleosome free regions seem to play a vital role in the formation of nearby nucleosomes, due to their high levels of sequence conservation and their association with transcription factors. Furthermore, well-positioned nucleosomes which flank the transcription start sites in the genome, also form a candidate responsible for imposing constraints in the organization of adjacent nucleosomes. This is based on a model that is similar to that of the 'parking lot' introduced by *Kiyama and Trifonov* [18].

The sequence preferences of these two types of nucleosome positioning (i.e. the sequences depleted of nucleosomes or Nucleosome Free Regions and the sequences with binded consistent nucleosomes or Nucleosome Binding Sites) are, therefore, more interesting to inspect in their respective sites, than those across the whole genome, due to their connections with several key cellular processes. These sequence elements are the center piece of this study, as we focus on providing evidence for the existence of constraints in this subset of nucleosomes that share particular positional and structural preferences.

3. PROPOSED APPROACH

This chapter elaborates about the possibilities how Machine Learning (ML) methods can help Genomic studies. Lately, more and more data is available spanning from bacterial genomes (E.coli) to single-cell eukaryotic genomes (S.cerevisiae) and even to complex multi-cell organism (H.sapiens), which facilitates the use of Machine Learning (ML) methods, in order to solve some everlasting problems fast and efficiently. In this chapter, we narrow further down how several Machine Learning methods can be applied on our problem, while also providing an overview and comparison between the different approaches that we've used.

3.1 Method Description

Nowadays, the field of Machine Learning is a very broad one, as it offers a vast range of algorithms, each one of which has been successfully applied in a wide range of fields, providing solutions for both the scientific and the industrial community. Spanning from Natural Language Processing and Anomaly Detection to Robotics and Bioinformatics, the field of Machine Learning is utilized in effectively solving numerous problems in a most efficient way.

In most cases, when we opt to use a machine to perform a task, we have to program it in order to accomplish it. However, in some cases, programming a machine to assist us in completing a task can be a lengthy and time-consuming choice. Yet, there exists a more optimal way to get things done, especially when enough data indicating how the program which potentially solves our problem should behave, exists. Generally, this is the main concept behind Machine Learning algorithms, which ML experts utilize in order to bridge that gap. Initially, a Machine Learning algorithm defines a model, which enables it to make predictions or judgements from the input data that we feed to it. It then proceeds to the phase of learning, in which the model parameters are optimized, based on given data or past experience. A more formal definition about this process can be found in the introductory book by Mitchell [19] :

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Supervised learning is a prominent branch of Machine Learning which challenges the problems it's facing by inferring (or "learning") a function from available data which has been labeled by human annotators. A common example of potential problems Supervised Learning methodologies tackle is that of deciding whether a patient's tumor is "malignant" or "not malignant", based on data from previous patients, the tumors of whom have been classified by the annotator as such.

The data used for the training process in supervised learning, comprising of labeled examples, is often called the "training set". Each example in the training set consists of an input value (generally in the form of a vector) and a label value, which is the desired output value corresponding to that vector. The algorithm utilizes the training data in order to infer a function which correctly reflects the data, by analyzing and finding patterns in it. The term "supervised" refers to the examples being labeled by the human user, who hence tells the learning machine what the right outcome of the inferred function is, depending on each example. This procedure opts to allow the machine to classify new unseen instances by assigning to them the correct output value, thus generalizing from the training data. These unseen examples are often referred to as the "test set".

Our problem is, therefore, a Supervised Learning use case as we opt to classify nucleotide sequences in two classes, the first being **Nucleosome Free Region** and the second being **Nucleosome Binding Site** - for further information please refer to chapter 1 - using the method of supervised learning, given two datasets of accordignly labeled data - one for each class - upon which we will train our classifiers.

3.2 Representations and Features

3.2.1 Vector Space Models

When problems of entity comparison occur, where entities can be sentences, documents or text corpora, a helpful strategy is that of mapping these entities into an indexing space where information may be more expressible and concise. A way of achieving that is by applying the given entities to a **vector space model**, a mathematical model that assists in mapping data into vectors. Initially aimed for information retrieval, the vector space models finally became a useful tool for machine learning, due to its ability of allowing the positioning of different types of items in a single type of space. Generally, a vector space model is applied on a set of given entities E_i by decomposing them into features f_j and mapping these features into a joint space. An analogous example is illustrated in Figure 3.1, where each instance has been mapped into a three-dimensional vector, each one composed from an extracted triplet of features. The dimensions may also be generalized to t, moving the representation to index spaces of higher dimension. Therefore, each entity E_i can be represented by a t-dimensional vector

$$E_i = (d_{i,1}, d_{i,2}, ..., d_{i,t}),$$

Where d_{ij} corresponds to the j-th feature.

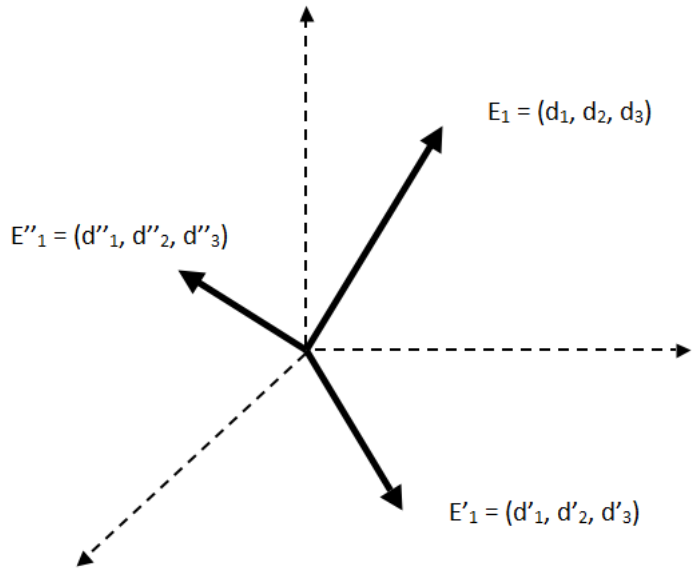

Figure 3.1: Vector representation of three-dimensional entity space

The next step towards an effective way of comparing two or more entities is to measure the similarity between them, expressed in $s(E_i, E_j)$ corresponding vectors. Usually, an efficient way of achieving this, is by defining a metric that reflects the degree of similarity between the vectors of each entity. A simple, yet powerful strategy towards this goal is to compute the inner product for each pair of vectors that we opt to compare, as the angle between them will usually suggest how similar or different are as it diminishes or grows, respectively.

3.3 Hidden Markov Models - HMMs

Hidden Markov Models (HMMs) have been the baseline statistical model used in sequential or temporal environments. Despite being limited, HMMs and their variants are the most widely used technique in that specific domain of studies, and are commonly regarded as the most popular one, due to their success when used in various systems and applications and general simplicity.

An HMM is nothing more than a probabilistic function of a Markov process. Here we provide a definition for Markov process (model).

A Markov process or Markov chain is a random process on which the Markov property can be applied. In other words, a Markov process can be described as a time sequence representing the evolution of a system represented by random variables that aren't independent, but rather the value of each variable depends on the previous states in the sequence. It is often quite reasonable to assume that one can make predictions for the future states of the process based solely on its present state as one could knowing the process's full full history, i.e. all the previous states plus the present state. Therefore, the evolution of a Markov process depends solely on the system's current state, entirely neglecting its future and past states.

For example, if the random variables measure the number of cars in a parking lot, then, knowing how many cars were parked in the parking lot today might adequately predict the number of cars that there will be parked tomorrow, without extra knowledge about the number of parked cars the parking lot had last week, let alone last year. That is, future states in the sequence are conditionally independent of past states, provided that we have information on the present state.

Every Markov process follows the Markov Assumption specified by Manning and Schuetze [20]:

"Suppose $X = (X_1, ..., X_T)$ *is a sequence of random variables taking values in some finite set* $S = s_1, ...s_N$*, the state space. Then the* Markov Properties *are :*

Limited Horizon:

$$P(X_{t+1} = s_k|X_1, ..., X_t) = P(X_{t+1} = s_k|X_t)$$

Time invariant (stationary):

$$P(X_2 = s_k|X_1)$$

Hence, we say that X is a Markov chain *or has the* Markov property."

A Markov chain can be, alternatively, represented by a state diagram as the one shown in Figure 3.2. The states correspond to the reports compiled by a hypothetical weather prediction center, including forecasts about sunny, rainy and cloudy days. According to the figure, a sunny day is followed by another sunny day 90% of the time, a cloudy day 7.5% of the time and a rainy day the other 2.5% of the time.

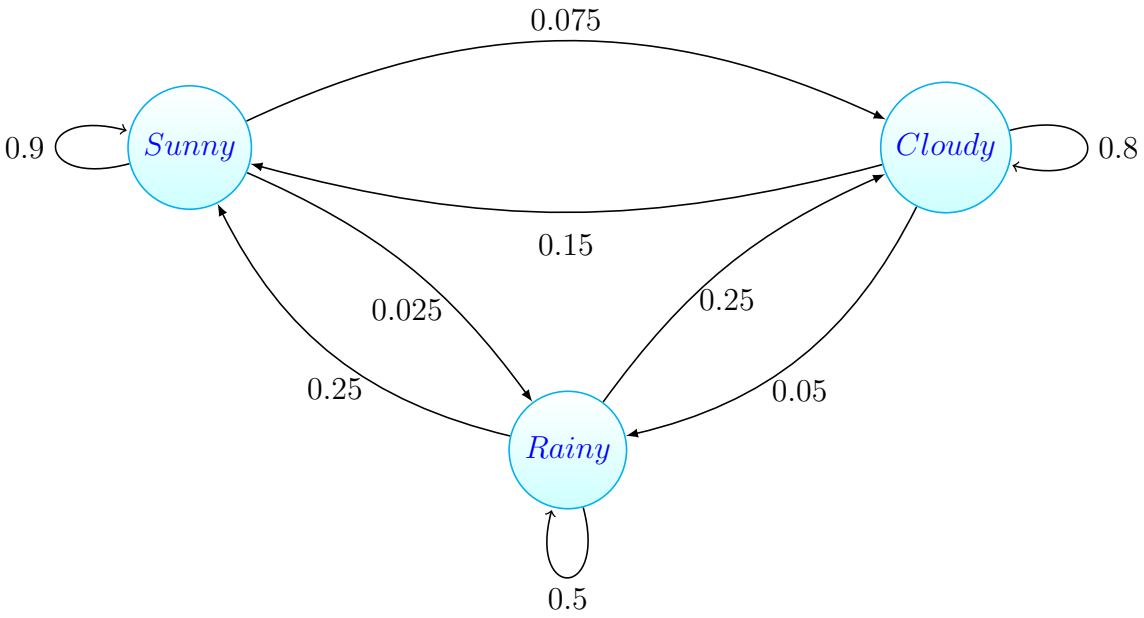

Figure 3.2: The above example.

Here, the circles represent the states, with the state name being at the center, while possible transitions between states are represented by arrows which connect them. Finally the edges are labeled with the probability of the transition from the state which lies at the tail of the arrow to the state at which the arrow points.

A notable observation which stems from the above example is that the sum of the probabilities of the outgoing edges is equal to 1. Moreover, it is emphasized that no long distance dependencies are present and the transition between one state to another is conditioned solely on the current state of the system, while being free of long distance dependencies.

Now that we have defined the Markov models, we can describe the Hidden Markov model:

A **hidden Markov model (HMM)** is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (*hidden*) states. However, despite not having knowledge about the state sequence that the model transitions through, that specific information is provided through a probability function which describes the evolution of the system.

Table 3.1: ***Notation used in the HMM chapter.***

Set of states	$S = \{s_1, ..., s_N\}$
Output alphabet	K = $\{\kappa_1, ..., \kappa_M\} = \{1, ..., M\}$
Intial state probabilities	$\Pi = \{\pi_i\}$, i $\in$ S
State transition probabilities	A = $\{a_{i,j}\}$,i,j $\in$ S
Symbol emission probabilities	B = $\{b_{i,j,k}\}$, i,j $\in$ S, k $\in$ K
State sequence	x = $(X_1, ..., X_{T+1})$ $X_t \rightarrow 1, ..., N$
Output sequence	O = $(o_1, ..., o_T)$ $o_t \in K$

The definition of a HMM can be based on a five-triple (S, K, Π, A, B) where the states and the output alphabet are symbolized with S and K respectively and the initial state probabilities, the transitions between states and the outputting of symbols are symbolized with Π, A and B respectively. We hypothesize that we cannot observe the states of the Markov process (or chain) at time t, hence $s(t)$ is hidden. The mapping from state names to integers which correspond to them is represented by the random variables X_t. Furthermore, there is a dependency between the symbol emitted at time and both the states at time t and at $t + 1$.

Hereby we provide a visualization of a Hidden Markov Process in the following example, which suits our approach:

Imagine that we are outside of a room without having vision of what is going on inside. In that room there are two buckets, labeled NFR and NBS each one of which contains a known mix of nucleotides, labeled T (for thymine), A (for adenine), C (for cytocine) and G (for guanine). Inside the room, someone chooses a bucket and randomly draws a nucleotide from it. He then proceeds by presenting the nucleotide to us in a way that can observe the sequence of the nucleotides coming out of the buckets, but not the sequence of buckets from which the nucleotides were drawn. Imagine that we are outside of a room without having vision of what is going on inside. In that room there are two buckets, labeled *NFR* and *NBS* each one of which contains a known mix of nucleotides, labeled *T* (for thymine), *A* (for adenine), *C* (for cytocine) and *G* (for guanine). Inside the room, someone chooses a bucket and randomly draws a nucleotide from it. He then proceeds by presenting the nucleotide to us in a way that can observe the sequence of the nucleotides coming out of the buckets, but not the sequence of buckets from which the nucleotides were drawn.

The person that's inside the room follows a certain procedure in choosing the buckets which lies on the fact that the choice of the bucket from which the n-th nucleotide will be drawn depends solely on the choice of the hat from which the $n - 1$-th nucleotide was drawn. This choice is not directly conditioned on the buckets chosen before the very bucket that was chosen previously. Therefore, the process followed by the person in the room has the Markov property. It can be described by the upper part of Figure 3.3, where NFR and NBS are the states of the model, T, A, C, G are the possible observations, the state transition probabilities are illustrated as numerical labels on arrows connecting the

states and the output probabilities appear as numerical labels on arrows connecting a state with an observation.

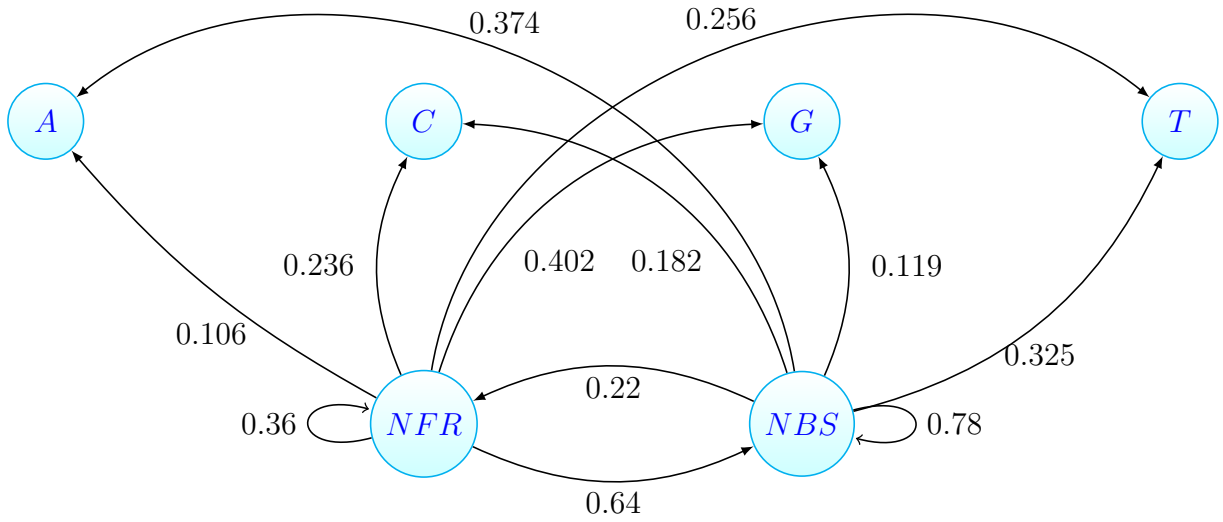

Figure 3.3: The Hidden Markov Model in our approach

In addition to the above, the Markov process is invisible to the observer, who sees only the sequence of nucleotides drawn from the buckets. This is the element responsible for calling this process a ”hidden Markov process”. Even if the observer has been informed about the composition of the buckets and has observed a sequence of nucleotides, he is not able to distinguish which bucket produced which nucleotide. However, the observer has other means to follow the process, such as knowing the likelihood that the n-th nucleotide shown came from each of the buckets.

HMMs are useful in that they efficiently describe surface events that are probabilistically generated by events which are undisclosed. In our approach, the HMMs can be useful in tagging parts of genomic sequences to being Nucleosome Free Regions or Nucleosome Binding Sites, taking into account the strict sequence of n-bases. We think of there being an underlying Markov chain of nucleotides from which the actual genomic sequences are generated.

FEATURES

We will initialize our work here by developing a feature vector with two dimensions, each one belonging to a class of our problem : *Nucleosome Free Region* - i.e a region depleted of nucleosomes - and *Nucleosome Binding Site* - a region with consistent nucleosomes - and then creating two Hidden Markov Models, one for each of the above classes. Thus, given a new sequence we will determine the probability of this sequence belonging to each one of our two classes. Hence, our feature vector will have its dimensions assigned to these two probabilities.

Here's a description of our feature vector v:

$$\begin{cases} v_{1i} = P(w_i \in A|w_i) \\ v_{2i} = P(w_i \in B|w_i) \end{cases}$$

where w_i describes the nucleotide sequences we want to classify, A is the set belonging to our first class and B is the set belonging to our second class. We can illustrate how the feature extraction works considering the following example :

Example 3.3.1. Consider two genomic sequences, **CGCAAATTATTG** and **AGCAAGACAGTT**, each one belonging to one of the above classes - we will use **NFR** and **NBS** for referring to the first and the second class respectively. We then proceed to compute the conditional probabilities of each genomic sequence belonging to each of the two classes, having the following results :

$$\begin{cases} P(\{CGCAAATTATTG\} \in \{NFR|CGCAAATTATTG\})=0.47 \\ P(\{CGCAAATTATTG\} \in \{NBS|CGCAAATTATTG\})=0.13 \end{cases}$$

and

$$\begin{cases} P(\{AGCAAGACAGTT\} \in \{NFR\}|AGCAAGACAGTT\})=0.25 \\ P(\{AGCAAGACAGTT\} \in \{NBS\}|AGCAAGACAGTT\})=0.62 \end{cases}$$

Hence, our feature vector for this two instances will be the following :

$$\begin{bmatrix} 0.47 & 0.13 \\ 0.25 & 0.62 \end{bmatrix}$$

We also included a different kind of feature extraction for the same HMM model, entitled **Normalized HMM**, previously used in *Antonakaki et. al* [21], where the feature vectors consist of the logarithm of the probability P of each input sequence, given the model, normalized by dividing it by the length l of the sequence (in nucleotides). More specifically our feature vector v will now have the following form:

$$\begin{cases} v_{1i} = log(P(w_i \in A|w_i))/l(w_i) \\ v_{2i} = log(P(w_i \in B|w_i))/l(w_i) \end{cases}$$

The reasons behind this approach are linked to the fact that the probability P is inversely proposal to the length, which means that, as the length of the input sequence increases, its probability decreases in an analogous manner resulting in unscaled data. Hence, we proceed to normalize that data using the logarithm function to achieve numerical stability and math simplicity and finally we normalize it, using the division by the sequence length described above. This may optimize the classification process by tackling any mismatches based on the previously unnormalized nature of the feature vectors.

3.4 Bag-of-words

The Bag-of-words model represents forms of text (such as sentences, documents or corpora) as the multiset of their words, which is called a **bag**. Despite keeping words which occur multiple times in a document, the Bag-of-words model ignores any form of linguistic structure present in the text, including grammar and word order. In most cases, a **bag of words** is a sparse vector which holds the number of times each word appears in the text. In our case, in order to create a bag-of-words (or, more precisely, a bag-of-nucleotides) model for a given genomic text document, we need to count the occurrence of each nucleotide in it.

We will now give an example by applying the bag-of-words model to a text document. Consider the following two sentences:

(2.3.1) Ryze is a good student. Annie is a good student too.

(2.3.2) Ryze is also good at playing football.
Based on these two sentences, we can construct a list which will hold the distinct words present in them:

$$[\text{"}Ryze\text{"},\text{"}is\text{"},\text{"}a\text{"},\text{"}good\text{"},\text{"}student\text{"},\text{"}Annie\text{"},\text{"}too\text{"},\text{"}also\text{"},\text{"}at\text{"},\text{"}playing\text{"},\text{"}football\text{"}]$$

Now we can represent each sentence by a 11-entry vector, using the indeces of the list that we've created:

$$[1, 2, 2, 2, 2, 1, 1, 0, 0, 0, 0] \tag{2.3.1}$$

$$[1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1] \tag{2.3.2}$$

Each scalar component of the above vectors represents the count of the corresponding entry in the list. For instance, the first three components of the first vector (which refers to the first sentence) are "1, 2, 2" because the first component, "Ryze" is encountered one time in it, the second component, "is", is encountered two times, while the third one, "a" is also encountered two times. Similarly, the first and second components of the second vector hold the value "1" because their corresponding words appear one time each in the second sentence, while the third component has the value "0" due to its absence from it.

Most of the times, in order to enhance the representational power of the bag-of-words model by simultaneously reducing its description length, we usually combine it with n-gram statistics. These hybrid models tend to reveal useful statistical information about the textual structure inside a document or between different documents. These models either generate each word based on a given number of preceding words which have already been encountered or neglect the word order entirely, by putting emphasis on word correlations which provide information about the topic of a document.

Although the most widely applied hybrid models are bigram-based – i.e. models which make predictions for each word by taking into account just the immediately previous one – there is flexibility in the available models one can use to harness the n-gram enhanced model. For example the length we choose for the n-grams may vary (2-grams, 3-grams and so on). In our approach we use the 3-gram model, or in our case, the 3-base.

The Bag-of-Words representation is of high importance to this study, because of the fact that it is independent of the order in which the nucleotides appear inside a sequence, meaning that it ignores the position of the n-base. This can possibly show that the positioning of nucleosomes is statistically correlated to the presence (or to the absence) of particular nucleotides or any forms of nucleotide n-grams in the underlying genomic sequence. It can also reinforce the fact that a genomic code which regulates genome positioning, despite being apparrent, doesn't actually depend on the sequence per se but rather on the nucleotide frequencies.

FEATURES

In this method, we will initialize our work by evaluating the frequency of each n-base - where $n \in \aleph$ - in our training set (the n is set by default to 3). Finally, we will proceed to construct two bags of 3-bases, one for each of our classes, which will act as a hash table the purpose of which will be to map each 3-base to its word count. Finally we will extract an initial feature vector for each class, consisting of the frequencies of the 3-bases belonging to that specific class. Hence, if the NFR and NBS bags have m and n discrete 3-bases respectively, our initial feature vectors v will have the following form :

$$v_i = f_i = N_i/m,\ where\ i \in \{0,\ ...,\ m\}$$

for the NFR class and :

$$v_i = f_i = N_i/n,\ where\ i \in \{0,\ ...,\ n\}$$

for the NBS class, where v_i is the i-th row of the feature vector, f_i is the frequency of the i-th 3-base in the corresponding class bag and N_i is the count of that particular 3-base in the class sequences.

Finally, we proceed to construct the final feature vector, which will then be used by the classication algorithms. We begin by defining a second temporary feature vector which consists of the frequencies of the 3-bases which form each testing instance and then we continue by computing the cosine similarity between this temporary feature vector and the initial feature vector described above. We will here provide a definition for cosine similarity :

Definition 3.4.1. Given two vectors of attributes, A and B, the cosine similarity, cos(θ), is represented using a dot product and magnitude as

$$similarity = cos(\theta) = \frac{A \cdot B}{\parallel A \parallel \parallel B \parallel} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_i^n A_i^2}\sqrt{\sum_i^n B_i^2}}$$

where A_i and B_i are components of vector A and B respectively.

Let us now give an example of the feature extraction that will take place :

Example 3.4.1. Consider the following eight genomic sequences, four of them belonging to the NFR class and four of them to the NBS class :

NFR	**NBS**
GCACT	AATCC
TGGCA	GAAAT
ACACT	TCCAT
CCACA	CATGT

By extracting the **overlapping** 3-bases we get the following bags :

NFR	**NBS**
–’GCA’,’CAC’,’ACT’˝	–’AAT’,’ATC’,’TCC’˝
–’TGG’,’GGC’,’GCA’˝	–’GAA’,’AAA’,’AAT’˝
–’ACA’,’CAC’,’ACT’˝	–’TCC’,’CCA’,’CAT’˝
–’CCA’,’CAC’,’ACA’˝	–’CAT’,’ATG’,’TGT’˝

The NFR and NBS bags will then be formed, by uniting the same-class 3-bases and computing the frequencies the distinct 3-bases :

NFR	**NBS**
–’GCA’ : 0.166˝	–’AAT’ : 0.166˝
–’CAC’ : 0.250˝	–’ATC’ : 0.083˝
–’ACT’ : 0.166˝	–’TCC’ : 0.166˝
–’TGG’ : 0.083˝	–’GAA’ : 0.083˝
–’GGC’ : 0.083˝	–’AAA’ : 0.083˝
–’ACA’ : 0.166˝	–’CCA’ : 0.083˝
–’CCA’ : 0.083˝	–’CAT’ : 0.166˝
	–’TGT’ : 0.083˝

Now let’s consider that we have the two instances of example 3.3.1. : **CGCAAATTATTG** and **AGCAAGACAGTT**. Their bags are the following :

CGCAAATTATTG	**AGCAAGACAGTT**
–’CGC’ : 0.1˝	–’AGC’ : 0.1˝
–’GCA’ : 0.1˝	–’GCA’ : 0.1˝
–’CAA’ : 0.1˝	–’CAA’ : 0.1˝
–’AAA’ : 0.1˝	–’AAG’ : 0.1˝
–’AAT’ : 0.1˝	–’AGA’ : 0.1˝
–’ATT’ : 0.2˝	–’GAC’ : 0.1˝
–’TTA’ : 0.1˝	–’ACA’ : 0.1˝
–’TAT’ : 0.1˝	–’CAG’ : 0.1˝
–’TTG’ : 0.1˝	–’AGT’ : 0.1˝
	–’GTT’ : 0.1˝

The reason we take the frequencies instead of the 3-base counts is simply to save time by skipping the division by $\| A \|$ and $\| B \|$.

Finally by computing the cosine similarity between the unlabeled sequences' bags and the two classes' bags we construct the final feature vectors.

The cosine similarity between the first testing instance bag and the NFR bag is :

$$0.1 \cdot 0.166 = 0.1166$$

while that between the first testing instance bag and the NBS bag is :

$$0.1 \cdot 0.083 + 0.1 \cdot 0.166 = 0.0249$$

With the same pattern we compute the cosine similarities between the second testing instance and the two class-bags resulting to the following feature vector :

0.1166	0.0249
0.0332	0

We also included a different kind of feature extraction for the same Bag-of-words model, entitled **Baseline Bag-of-words**, in which we used the raw counts of each 3-base per instance as features. More specifically, our language L will consist of all the 3-bases produced over the alphabet $\{A, C, G, T\}$, therefore $|L| = 4^3 = 64$. This results in the following 64-dimensional feature vectors :

$$v_{i,j} = N_j, \ where \ i \in \{1, ..., n\}, j \in \{1, ..., 64\}$$

where $v_{i,j}$ is the i-th feature vector, n is the total number of instances and N_j is the count of the j-th 3-base for the corresponding instance.

Let us now give an example of the feature extraction that will take place, based on the sequences *CATCATAATCGGTTC* and *ATTAATTAGGGGGCA* of the previous example :

Example 3.4.2. To construct the feature vectors for our NFR and NBS classes, we just have to compute the count of each distinct 3-base, for each of our instances. Hence, the following feature vectors will be constructed:

CGCAAATTATTG	**CGCAAATTATTG**
–'AAA' : 1″	–'AAA' : 0″
–'AAC' : 0″	–'AAC' : 0″
⋮	⋮
–'AAT' : 1″	–'AAT' : 0″
⋮	⋮
–'ACA' : 0″	–'ACA' : 1″
⋮	⋮
–'ATT' : 2″	–'ATT' : 0″
⋮	⋮
–'CAA' : 1″	–'CAA' : 1″
⋮	⋮
–'CAG' : 0″	–'CAG' : 1″
⋮	⋮
–'GCA' : 1″	–'GCA' : 1″
⋮	⋮
–'TAT' : 1″	–'TAT' : 0″
⋮	⋮
–'TTG' : 1″	–'TTG' : 0″
⋮	⋮

This type of feature extraction clearly results in sparse multidimensional feature vectors of dimension $d = 64$. The reason behind this specific approach is to indicate the effect of *dimensionality reduction* to our problem by making a comparison between this 64-dimensional model with the 2-dimensional one, described at the beginning of this section.

3.5 N-gram Graphs

Another useful model which can be used to represent text data is the n-gram graph. In text-related studies, an n-gram is a set which consists of n characters or words. The n-gram graph model has been successfully applied to a variety of disciplines, including text summarization [22], classification [23] and sequence clustering.

Table 3.2: N-gram examples from various disciplines.

Field	Unit	Sample sequence	1-gram	2-gram	3-gram
Protein sequencing	**amino acid**	. . . Cys-Gly- -Leu-Ser-Trp...	. . . , Cys, Gly, Leu, Ser, Trp,...	. . . , Cys-Gly, Gly-Leu, Ser-Trp, ...	. . . , Cys-Gly-Leu, Gly-Leu-Ser, Leu-Ser-Trp, . . .
DNA sequencing	**base pair**	...AGCTTCGA...	..., A, G, C, C, T, T, C, G, A, . . .	..., AG, GC, CT, TT, TC, CG, GA, . . .	..., AGC, GCT, CTT, TTC, TCG, CGA, . . .
Computational linguistics	**character**	...to˙be˙or˙ not˙to˙be. . .	. . . , t, o, ˙, b, e, ˙, o, r, ˙, n, o, t, ˙, t, o, ˙, b, e, . . .	. . . , to, o˙, ˙b, be, e˙, ˙o, or r˙, ˙n, no, ot, t˙, ˙t, to, o˙, ˙b, be, . . .	. . . , to˙, o˙b, ˙be, be˙, e˙o, ˙or, or˙, r˙n, ˙no, not, ot˙, t˙t, ˙to, to˙, o˙b, ˙be, . . .
Computational linguistics	**word**	. . . to be or not to be . . .	. . . , to, be, not, to, be, . . .	. . . , to be, be or, or not, not to, to be, . . .	. . . , to be or, be or not, or not to, not to be, . . .

An n-gram graph can be constructed by following three basic methods, depending on the ways in which the adjacency between neighboring n-grams is defined. Generally, we can use a window of a given character width which surrounds the n-gram of interest N_0 and includes the characters which are regarded as being the neighbors of it. Each neighborhood present in the original text corresponds to a pair of connected vertices in the n-gram graph which represent the neighboring n-grams. Here we provide a definition for the symmetric approach, which we follow in this study, as provided by Giannakopoulos et al.[22].

The symmetric approach - *"A window of length n runs over the summary text, centered at the beginning of N_0. If the n-gram we are interested in is located at position p_0, then the window will span from $p_0 - [\frac{n}{2}]$ to $p_0 + [\frac{n}{2}]$, taking into account both preceding and following characters or words. Each neighbourhood indicative edge is weighted based on the number of window co-occurrences of the neighbours, as previously indicated, within the text."*

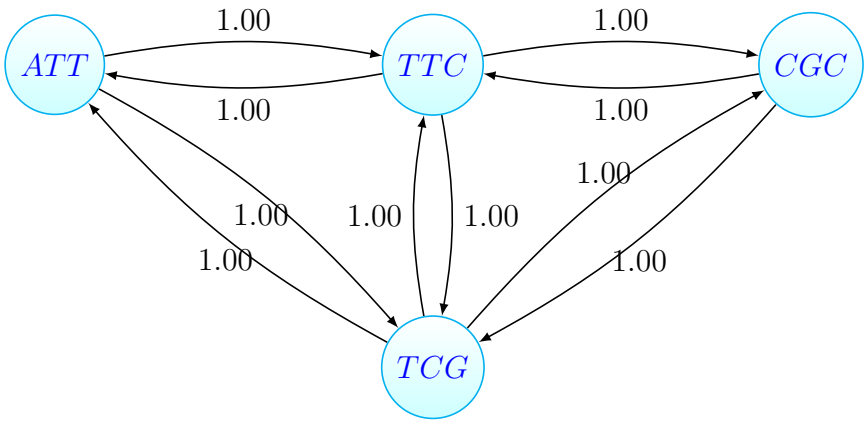

Figure 3.4: Graph extracted from the genomic sequence ATTCGC based on the symmetric type of window.

We now provide the definition of n-gram, given a text, as it is described by *Giannakopoulos et al.* [22]:

Definition 3.5.1. *If $n > 0$, $n \in \mathbb{Z}$, and c_i is the i-th character of an l-length character sequence $T^l = \{c_1, c_2, ..., c_l\}$ (our text), then a character n-gram $S^n = (s_1, s_2, ..., s_n)$ is a subsequence of length n of $T^l \iff \exists i \in [1, l-n+1] : \forall j \in [1, n] : s_j = c_{i+j-1}$. We shall indicate the n-gram spanning from c_i to c_k, $K > i$, as $S_{i,k}$, while n-grams of length n will be indicated as S^n.*

The above definition means that n-gram S_n can be located in the original text in the form of a substring of length n, which begins at the i-th and ends at the j-th character of the original text. For example if the text is the following genomic sequence:

GCACTTCATTGCTATCCAACGTTACATTAGCGG

then $S_{1,2}$ is the sequence 'G','C'≡'GC' for display purposes.

Extracting N-grams
If we choose to extract the n-grams (S^n) of a text T^l, the (elementary) algorithm is indicated as algorithm 1.

Input: text
Output: n-gram set
//`T is the text we analyse`
for *all i in [1, length(T)-n+1]* **do**
| get substring of T from index i to i+n-1
end

Algorithm 1: Extraction of n-grams

Now we can provide an example of this algotithm's effect on genomic textual data :

Example 3.5.1. Application of our method to the genomic sequences **CGCAAATTATTG** and **AGCAAGACAGTT** we have used in examples 3.3.1, 3.4.1, with a requested n-gram size of 3 would return:

–'CGC', 'GCA', 'CAA', 'AAA', 'AAT', 'ATT', 'TTA', 'TAT', 'ATT', 'TTG˝ and
–'AGC', 'GCA', 'CAA', 'AAG', 'AGA', 'GAC', 'ACA', 'CAG', 'AGT', 'GTT˝

for **CGCAAATTATTG** and **AGCAAGACAGTT** respectively, while an algorithm taking disjoint n-grams would return:

–'CGC', 'AAA' 'TTA', 'TTG˝ and –'AGC', 'AAG', 'ACA', 'GTT˝

for **CGCAAATTATTG** and **AGCAAGACAGTT** respectively.

N-gram Graph

The *n-gram graph* is a graph $G = \{V^G, E^G\}$ the vertices of which are n-grams $v^G \in V^G$ and the edges $e^G \in E^G$ which connect them represent the adjacency between the corresponding n-grams (also see figure 3.4). The edges usually carry weights which indicate the distance between the two n-gram neighbors in the text. A more formal definition is given by Giannakopoulos et al. [22]:

Definition 3.5.2. *"If $S = \{S_1, S_2, ...\}, S_k \neq S_l, \forall k \neq l, k, l \in \mathbb{N}$ is the set of distinct n-grams extracted from a text T^l, and S_i is the i-th extracted n-gram, then G is a graph, where there is a bijection (one-to-one and onto) function $f : S \to V$."*

The edges E, are labeled with weights c_{ij} where c_{ij} corresponds to the number of times an adjacency of a specific pair of n-grams S_i, S_j grams occurs within a distance D_{win} (is commonly measured in characters) of each other.However, in order to reduce the complexity of the representation and due to the diminishing importance of a relation between n-grams neighboring each other while the distance between them increases, one has to consider only a window surrounding S_i in the original text, in order to deduce which neighbors S_j are worth examining. The adjacency between two given neighboring n-grams S_i, S_j which happen to be located inside the window D_{win} and are represented by vertices v_i, v_j , is representing by the corresponding edge $e \equiv \{v_i, v_j\}$ that connects them.

All in all, an n-gram graph is a graph which contains n-grams extracted from textual data (or in our case genomic textual data) and bears context-related information. By using the n-gram graph as a representation in this study we are trying to augment our undestanding of genomic text data not only by capturing the simple co-occurence of nucleotides or n-bases (i.e. n-grams of nucleotides), but also by allowing for different types of the same n-base. This may reinforce the fact that nucleosome positioning is not regulated by the exact order in which nucleotides appear inside a genomic sequence, but rather it depends on the presence (or absence) of certain nucleotides or n-bases in the underlying genomic sequence. It also may indicate whether neighborhoods of n-bases can identify NFR or NBS genomic sequences, taking into account the relative positioning of n-bases.

FEATURES

We will initialize our work here by constructing two n-gram graphs based on the symmetric approach, one for each of our classes. This can be done by merging the n-gram graphs of the genomic sequences belonging to the training instances of the same class - let G_A be the n-gram graph of the first class (i.e. NFR) and G_B the n-gram graph of the second class (i.e. NBS). In order to keep these graphs balanced (i.e. each n-gram graph that

gets merged to it has the same effect on the final resulting graph), the class n-gram graph is scaled by the amount of n-grams that are so far merged into it. A definition of the union between graphs G_1 and G_2 is given formally in: [24]

Definition 3.5.3. *"If E^1 and E^2 are the edge sets of G_1, G_2 correspondingly, W_i is the result graph edge weighting function and W_1, W_2 are the weighting function of the operand graphs with $e \notin E^i \implies W_i(e) = 0, i \in 1, 2$, then the edgeset E of the merged graph is*:

$$E = E^1 \cup E^2, W^i(e) = \frac{W^1(e) + W^2(e)}{2}, e \in (E^1 \cup E^2)$$

Subsequently we proceed by constructing an n-gram graph for each testing genomic sequence w_i which will then be compared with each of the two class graphs mentioned above. The similarities which will be extracted by this comparison can be assigned to the dimensions of our feature vector.

In this study we use three forms of similarity metrics between two n-grams G^i, G^j: the Value Similarity (or VS) which measures the amount of edges which are contained both in graph G^i and in graph G^j, the Size Similarity (or SS) which compares the size between G^i, G^j and the Containment Similarity (or CS) which measures the part of G^i contained by G^j. Here we will provide formal definitions fo each metric as given formally in [24]:

Definition 3.5.4.

$$VS(G^i, G^j) = \frac{\sum_{e \in G^i} \frac{min(w_e^i, w_e^j)}{max(w_e^i, w_e^j)}}{max(|G^i|, |G^j|)}$$

Definition 3.5.5.

$$SS(G^i, G^j) = \frac{min(|G^i|, |G^j|)}{max(|G^i|, |G^j|)}$$

Definition 3.5.6.

$$CS(G^i, G^j) = \frac{\sum_{e \in G^i} \frac{min(w_e^i, w_e^j)}{max(w_e^i, w_e^j)}}{|G^i|}$$

Hence, our vector v will be:

$$\begin{cases} v_{i1} = VS(G_{w_i}, G_A) \\ v_{i2} = VS(G_{w_i}, G_B) \\ v_{i3} = SS(G_{w_i}, G_A) \\ v_{i4} = SS(G_{w_i}, G_B) \\ v_{i5} = CS(G_{w_i}, G_A) \\ v_{i6} = CS(G_{w_i}, G_B) \end{cases}$$

where G_{w_i} is the n-gram graph extracted from the genomic sequence test instance w_i.

3.6 Classification Algorithms

3.6.1 Decision Trees

We will here give a brief description of decision trees. For a more complete overview the reader can check either [25] or [26].

When we want to say that an instance belongs to a certain class we sometimes subject it to a series of questions upon its attributes or features. On receiving an answer to these question, we follow up with another, until we can carefully distinguish the class to which the instance in question belongs. This sequence of questions can be organized in a decision tree, a hierarchical structure consisting of nodes and directed edges. A decision tree has three types of nodes:

- The **root node**, which hasn't got any internal edges and zero or more external edges.
- The **internal nodes**, each one of which has exactly one internal edge and two or more external edges.
- The **leaves** or **terminal nodes**, each one of which has exactly one internal edge and one external edge.

Figure 3.5 provides an illustration of a decision tree for the training set of Figure 3.6.

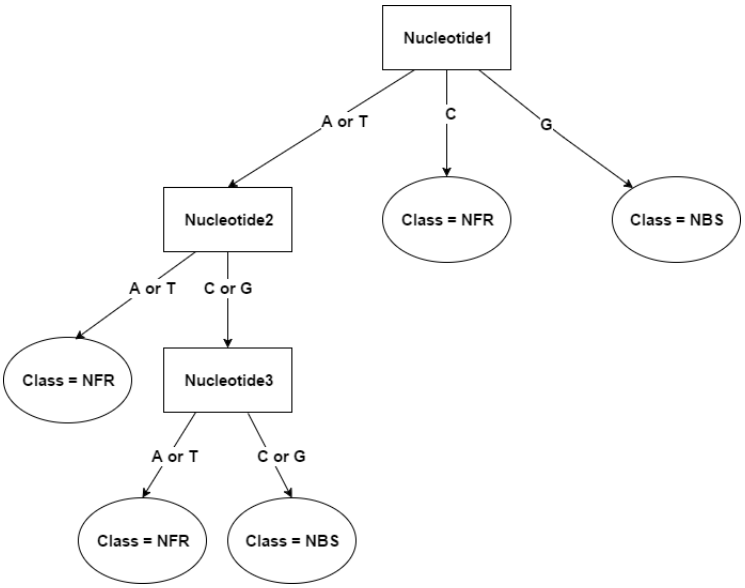

Figure 3.5: A decision tree

Nucleotide1	Nucleotide2	Nucleotide3	Class
A	C	G	NBS
C	G	T	NFR
A	A	T	NFR
T	T	C	NFR
C	G	A	NFR
G	A	T	NBS
G	C	G	NBS
T	T	A	NFR

Figure 3.6: Training set

In a decision tree, a class label is assigned to each leaf node. The **non-terminal nodes**, which consist of the root node and the internal nodes, contain decision conditions in order to distinguish the instances that have different attributes or features. If we try to classify, for example, the instance "ACG" (the first from our training set in Figure 3.6) using the decision tree from Figure 3.5, then the classification process for this instance will begin from the root node **Nucleotide1**, proceed to the internal node **Nucleotide2** and finally reach the terminal node **Nucleotide3** where it will be classified as belonging to the NBS class. Algorithm 2 presents a general pseudo-code for building decision trees.

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the
classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root best
        for each value v_i of best do
            examples_i ← elements of examples with best = v_i
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Algorithm 2: Pseudo-code for building a decision tree

3.6.2 Naive Bayes

In some cases the connection between the instance characteristics and the class variable is non-deterministic. In other words, the class label of a given instance can't be predicted with certainty, even when the attributes of the instance resemble very much those of the already classified training instances. This uncertainty can be modeled with the assistance

of the Bayes Theorem:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

where X and Y are random variables. For example, X might be a random variable representing the fact that a genomic sequence contains the trinucleotide "ATT" and Y might be a random variable representing the fact that this genomic sequence is a **Nucleosome Free Region** (or a *NFR*).

Hence, let's suppose that **X** is used to define the set of characteristics and **Y** is used to define the class variable. If those two variables are correlated in a deterministic way, then it is possible to utilize **X** and **Y** as random variables, in order to define their probabilistic relationship using the posterior probability $P(Y|X)$. This is the probability of **Y** happening, provided that we have evidence of **X** happening.

The Bayes theorem is, therefore, useful in the way that it allows us to define the posterior probability in relation to the prior probability $P(Y)$, the class-conditional probability $P(X|Y)$ and the evidence $P(X)$.

The Naive Bayes classifier makes the assumption that the characteristics are conditionally independent, considering a given class y, in order to be able to compute the class-conditional probability. This assumption can be expressed as:

$$P(X|Y = y) = \prod_{i=1}^{d} P(X_i|Y = y),$$

where we have d characteristics for each instance of the training set.

Therefore, in order to classify each instance in the test set, the Naive Bayes classifier computes the posterior probability for every class Y:

$$P(Y|X) = \frac{P(Y)\prod_{i=1}^{d} P(X_i|Y = y)}{P(X)}$$

and chooses the class Y which maximizes the numerator of the above equation, since the denominator P(X) will always be constant.

3.6.3 k-Nearest Neighbor

Sometimes, when we want to assign a class label to an instance we choose to compare its characteristics with those belonging to the training set instances. For the classification process to be as accurate as possible we need to find the examples in the training set, the characteristics of which tend to resemble those of the instance to be classified. These examples are often known as the **nearest neighbors** and can be utilized in order to define the correct class label for the test instance.

A nearest neighbor classifier represents every instance as a point in a d-dimentional

space, where d is the number of characteristics we have for each instance. Given a new unclassified instance, the classifier can measure its proximity to the training set instances, by mapping it at this d-dimentional space and then computing the distance between it and the rest of the data using a proximity measure. Hence, given an unclassified instance z, we use the term k-nearest neighbors to refer to the k points in the data that are closer to z. In Algorithm 3, a pseudo-code example for the instance base learning methods is illustrated.

```
1) for each testing instance
    1.1) find the k nearest instances of
    the training set according to a
    distance metric
    1.2) Resulting Class = most frequence class
    label of the k nearest instances
```

Algorithm 3: Instance-based learning pseudo-code

If the value of k is too little, then the nearest neighbor classifier might be susceptible to underfitting due to the presence of noise in the training set. On the other hand, if the value of k is too large, the classifier might be error-prone in its classification outcome, due to the possibility that the nearest neighbors may include points in space which aren't situated in close proximity to the unclassified instance. Lastly, in the case that the neighbors hold more than one labels, the label assigned to the majority of the neighbors is then assigned to the unclassified instance. This is shown in the next equation:

$$y' = argmax_v \sum_{(x_i,y_i)\in D_z} I(v = y_i)$$

, where v is a class label, y_i is a class label belonging to the i-th neighbor and $I()$ is a function which returns the value of 1 when its argument is true and 0 otherwise.

As we see in the above equation, every neighbor has equal saying on the outcome of the classification, although their distance from the unclassified data point may vary greatly. Therefore, a way of maintaining a safe classification criterion is to normalize the influence that each neighbor has on the class label, in relation to its distance from the test instance: $w_i = 1/d(x', x_i)^2$. By doing this we allow the neighbors which are situated closer to the unclassified data point to have much greater influence on the classification outcome than the ones which tend to be found at a greater distance from it.

3.6.4 Support Vector Machines

A classification technique which has gained considerable attention, is the Support Vector Machine (SVM). This particular technique, the roots of which lie in statistical training theory, has shown many promising results in many cases that it has been applied on. The main reason for that is the efficiency of the SVM in handling multi-dimentional data,

which gives it the ability to tackle the curse of dimensionality. Another important aspect of this classification technique is that it makes decisions based on a subset of the training instances, known as the **support vectors**.

The SVM initiates by representing the training examples in a feature space, mapping them in a way that the points belonging to different categories are separated by a gap. Then, during the classification process, the SVM maps the new unseen examples into that same space, in order to predict the category to which they belong to. Therefore, the separation gap must be as wide as possible, in order for the SVM to be more accurate in its predictions.When the training instances can't be separated linearly, the SVM is also able to perform an efficient non-linear classification process, by mapping the training examples into feature spaces which are high-dimensional. Hence, the Support Vector Machine classifier opts to define the hyperplane for which the largest separation, or margin, between the data points of the different classes in our data, can be achieved. In other words, the aim of this classiication technique is to locate which feature space allows for the maximum distance possible between the nearest points in our data belonging to different classes. This hyperplane, if exists, is commonly known as the **maximum-margin hyperplane**.

A general pseudo-code for SVMs is illustrated in Algorithm 4.

1) Introduce positive Lagrange
multiplers, one for each of the
inequality constraints. This
gives Lagrangian:

$$L_p = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} a_i y_i (x_i \cdot w - b) + \sum_{i=1}^{N} a_i$$

2) Minimize L_p with respect to w, b.
This is a convex quadratic programming problem.
3) In the solution, those points
for which $a_i > 0$ are called "support vectors"

Algorithm 4: SVM pseudo-code

where **w** is termed the weight vector and b the bias which are used in order to define the decision rule for the algorithm given by:

$$f_{\mathbf{w},b}(\mathbf{x}) = sgn(\mathbf{w}^T\mathbf{x} + b)$$

3.7 Evaluation

3.7.1 Evaluation Metrics Definition

The results produced from various classification tasks can be represented in confusion matrices [27] such as the one shown in Table 3.1. The purpose of this matrix is to represent the number of instances that were classified correctly (True positives or TP and True Negatives or TN) and falsely (False Positives or FP and False Negatives or FN). In Table 3.1 we provide a confusion matrix for a binary classification task, where n is the number of instances to be classified.

n = 150	**Predicted Value**: *N*	**Predicted Value**: *P*
Actual Value: *N*	True Negatives (or *TN*): 45	False Positives (or *FP*): 15
Actual Value: *P*	False Negatives (or *FN*): 10	True Positives (or *TP*): 80

Figure 3.7: Confusion matrix

The confusion matrix aids us in computing several performance metrics, in order to accurately evaluate our models.
We are going to present some performance metrics, of which we will use the first four in our experiments:

- **Accuracy**
$$Accuracy = \frac{TP + TN}{N}$$
- **Recall**
$$Recall = \frac{TP}{TP + FN}$$
- **Precision**
$$Precision = \frac{TP}{TP + FP}$$
- **F-measure**
$$F - measure = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision}$$
- **Specificity**
$$Specificity = \frac{TN}{FP + TN}$$

- **AUC**

$$AUC = 2 \cdot Accuracy \cdot Precision$$

3.7.2 What do different metrics mean ?

In this section we are going to analyze the meaning of each measure. We will define models A, B and C to facilitate our examples. Here follow the Confusion Matrices of each model :

	Predicted Value: *N*	**Predicted Value**: *P*
Actual Value: *N*	True Negatives (or *TN*): 0	False Positives (or *FP*): 0
Actual Value: *P*	False Negatives (or *FN*): 60	True Positives (or *TP*): 90

Figure 3.8: Confusion matrix of model A

	Predicted Value: *N*	**Predicted Value**: *P*
Actual Value: *N*	True Negatives (or *TN*): 60	False Positives (or *FP*): 90
Actual Value: *P*	False Negatives (or *FN*): 0	True Positives (or *TP*): 0

Figure 3.9: Confusion matrix of model B

	Predicted Value: *N*	**Predicted Value**: *P*
Actual Value: *N*	True Negatives (or *TN*): 10	False Positives (or *FP*): 15
Actual Value: *P*	False Negatives (or *FN*): 50	True Positives (or *TP*): 75

Figure 3.10: Confusion matrix of model C

- **Accuracy**
  The accuracy metric measures how efficient is our model at correctly classifying instances. When evaluating models, we need to ensure that we pick the model with the highest accuracy.

  □ The accuracy of the A model is (90/150)*100 or 60%.

- The accuracy of the B model is (60/150)*100 or 40%.
- The accuracy of C is (85/150)*100 or 56.7%.

The accuracy measure suggests that A is a better model while C comes second with a close margin. B is the far worst model.

- **Recall**
  The recall metric is defined as the division between the number of accurate positive predicted instances and the number of the instances belonging to the positive class in our data. This metric measures how complete a classifier is. A low recall indicates that many instances in our data are classified as being negative class instances despite belonging to the positive class.

  - The recall of the A model is 90/150*100 or 60%.
  - The recall of the B model is 0/(0+60)*100 or 0%.
  - The recall of C is 75/(75+50)*100 or 60%.

  The recall metric suggests C is a better model and that the B is more useful than the A model even though it has a lower accuracy. The difference in recall between the B model and the C can be explained by the large number of False Positives predicted by the B model.

- **Precision**
  The precision metric is defined as the division between the number of accurate positive predictions and the total number of the instances which were predicted as belonging to the positive class in our data. This metric measures how exact a classifier is. A low precision indicates that many instances in our data are classified as being positive class instances despite belonging to the negative class.

  - The precision of the A model is 90/(90+0)*100 or 100%.
  - The precision of the B model is 0/(0+90)*100 or 0%.
  - The precision of C is 75/(75+15)*100 or 83.4%.

  The precision suggests C is a better model and that the B is more useful than the A model even though it has a lower accuracy. The difference in precision between the B model and the C can be explained by the large number of False Positives predicted by the B model.

- **F-measure**
  The F-measure is also called the F Score or the F Measure. This metric measures how balanced are the metrics Recall and Precision for a given model.

  - The F-measure of the A model is 2*(0.6*1.0)/(0.6+1.0)*100 or 75%.
  - The F-measure of the B model is 2*(0.0*0.0)/(0.0+0.0)*100 or undefined.
  - The F-measure of C is 2*(0.6*0.834)/(0.6+0.834)*100 or 69.8%.

> If we were looking to select a model based on a balance between precision and recall, the F-measure suggests that B model is the one to beat and that C model is not yet sufficiently competitive.

This example illustrates the fact that the accuracy measure can be misleading. Hence, it is often more preferable to select a model which has large precision and recall despite having lower accuracy, because of its ability of producing correct predictions for our problem.

For example, when we have a problem of class imbalance (i.e. a problem where the number of the instances belonging to one class is many times larger than the number of the instances belonging to the other), the model will usually predict the majority of the instances as belonging to the ”major” class in a rather naive way, by just simply taking into account the class imbalance rather than being accurate by extracting useful representations from the data.

4. EXPERIMENTS

4.1 Experimental Setup

The dataset we will use for our study consists of the the *S. cerevisiae* genome and is the similar dataset to that used in [17]. Due to it being a commonly used dataset in previous studies, we can easily compare our results with that of previous experiments.

Here, follows a description of the data files :

- **.bed files**
  These .bed files (browser extensible data) have a quite simple structure, which consists from three elements : a) the chromosome to which the specific data belongs, b) the starting point and c) the ending point of it.
  For example the line "*chr5 100 200*" means that the element we've encountered belongs to the 5-th chromosome from the 100-th position to the 200-th.

- **.fa files**
  These .fa files complement the above .bed files by providing more information for the specific coordinated elements, due to having not only the information about the three elements described above, but also having information about the nucleotide sequences. Hence, the above example now looks like this: "*chr5:100-200 ATGAGA...*"

Our experiments were ran in a MacBook Air (13-inch, Early 2014) featuring a 1.4GHz dual-core Intel Core i5 (with Turbo Boost up to 2.7GHz) with 3MB shared L3 cache. Since our code does not take advantage of both cores, the processor possibly uses just one core with the latter clock frequency.

For handling the Hidden Markov Model part of the implementation, Jahmm[1], a Java library implementing the various algorithms related to HMMs was used. For training our HMM model, we used the Baum-Welch algorithm [28], which was implemented in this library. Jahmm's original author is Jean-Marc Francois.

In addition, for the n-gram graph part of the implementation, the JINSECT[2] toolkit was

[1] https://github.com/aubry74/Jahmm
[2] https://sourceforge.net/p/jinsect/wiki/Home/

used. JINSECT is a Java-based toolkit and library that supports and demonstrates the use of n-gram graphs within Natural Language Processing applications. For our implementation, we took advantage of JINSECT's tools capable of creating, merging and comparing n-gram graphs, which facilitated the feature extraction needed for the experiments. JINSECT was written by George Giannakopoulos and Panagiotis Giotis.

Finally, we used Weka[3] [29] a collection of machine learning algorithms for data mining tasks, in order to utilize these algorithms for our classification tasks. The Weka Data Mining Software was implemented by the Machine Learning Group at the University of Waikato.

4.2 Results

We hereby present our experimental results. We used the measures Recall, Precision and F-measure for our experiments and we also computed the Mean Training Time per Algorithm/Representation Pair, the Mean Classification Time per Algorithm/Representation Pair and the Mean Total Time per Algorithm/Representation Pair.

[3]http://www.cs.waikato.ac.nz/ml/weka/

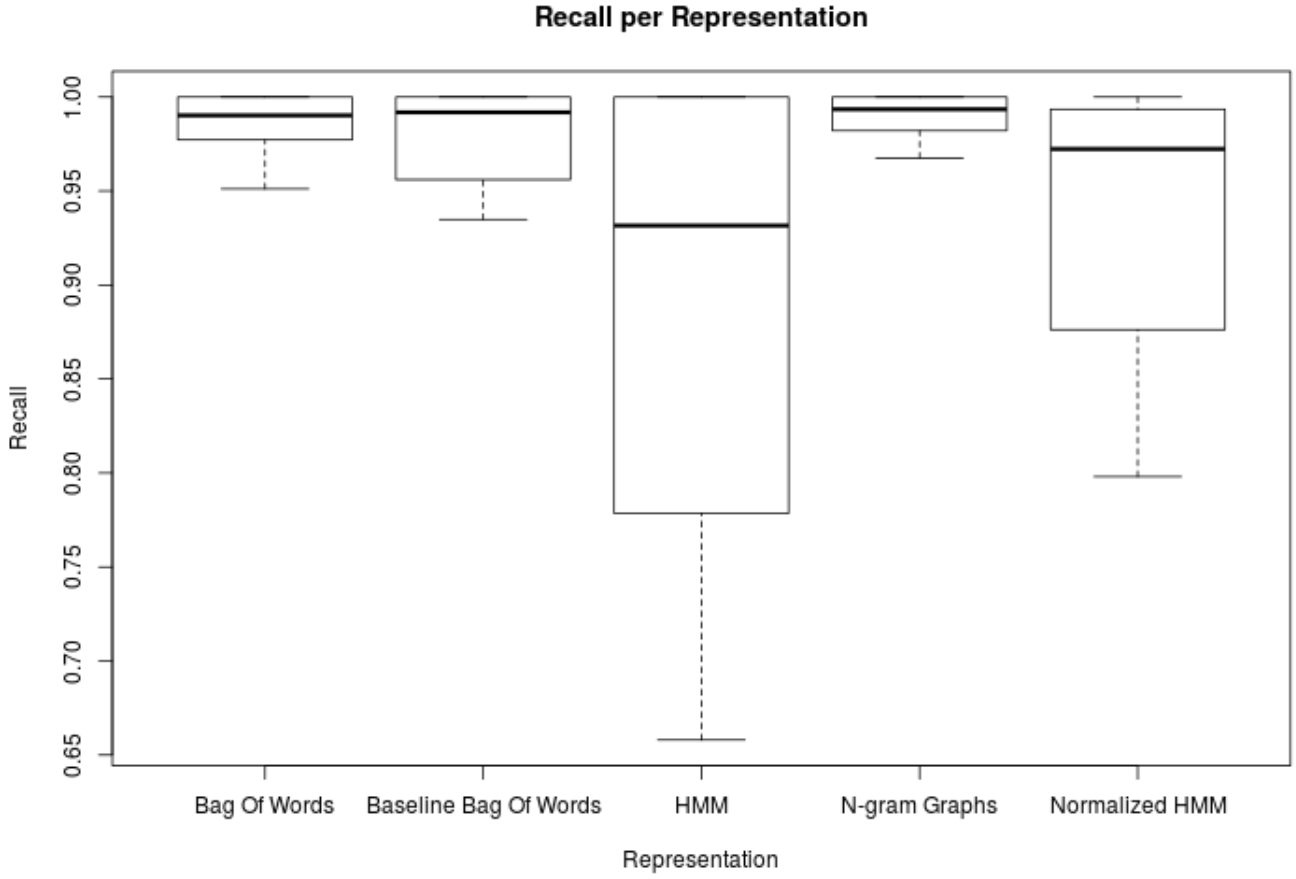

Figure 4.1: Recall measure per Representation (values between 0.00 - 0.65 neglected due to lack of data).

The most successful models in terms of Recall are: the N-gram graphs and Baseline Bag-of-words models the mean Recall of which stands at 0.990 and 0.985 respectively. HMMs tend to be the least successful ones featuring a mean Recall of 0.884. Normalized HMMs appear to be more effective than plain ones with a mean Recall at 0.940, while the Bag-of-words model featuring dimensionality reduction tends to be less effective than its Baseline model by a mean margin of 0.007.

Table 4.1: Mean Recall per Algorithm/Representation Pair (higher is better).

Representation	Classification Algorithm	Mean Recall
HMM	Naive Bayes	0.713
	Decision Trees	**1.000**
	kNN	0.824
	SVM	**1.000**
Normalized HMM	Naive Bayes	0.952
	Decision Trees	0.976
	kNN	0.834
	SVM	0.990
Bag-of-words	Naive Bayes	0.987
	Decision Trees	0.989
	kNN	0.965
	SVM	0.978
Baseline Bag-of-words	Naive Bayes	**1.000**
	Decision Trees	0.945
	kNN	0.970
	SVM	**1.000**
N-gram graphs	Naive Bayes	0.991
	Decision Trees	0.993
	kNN	0.977
	SVM	**1.000**

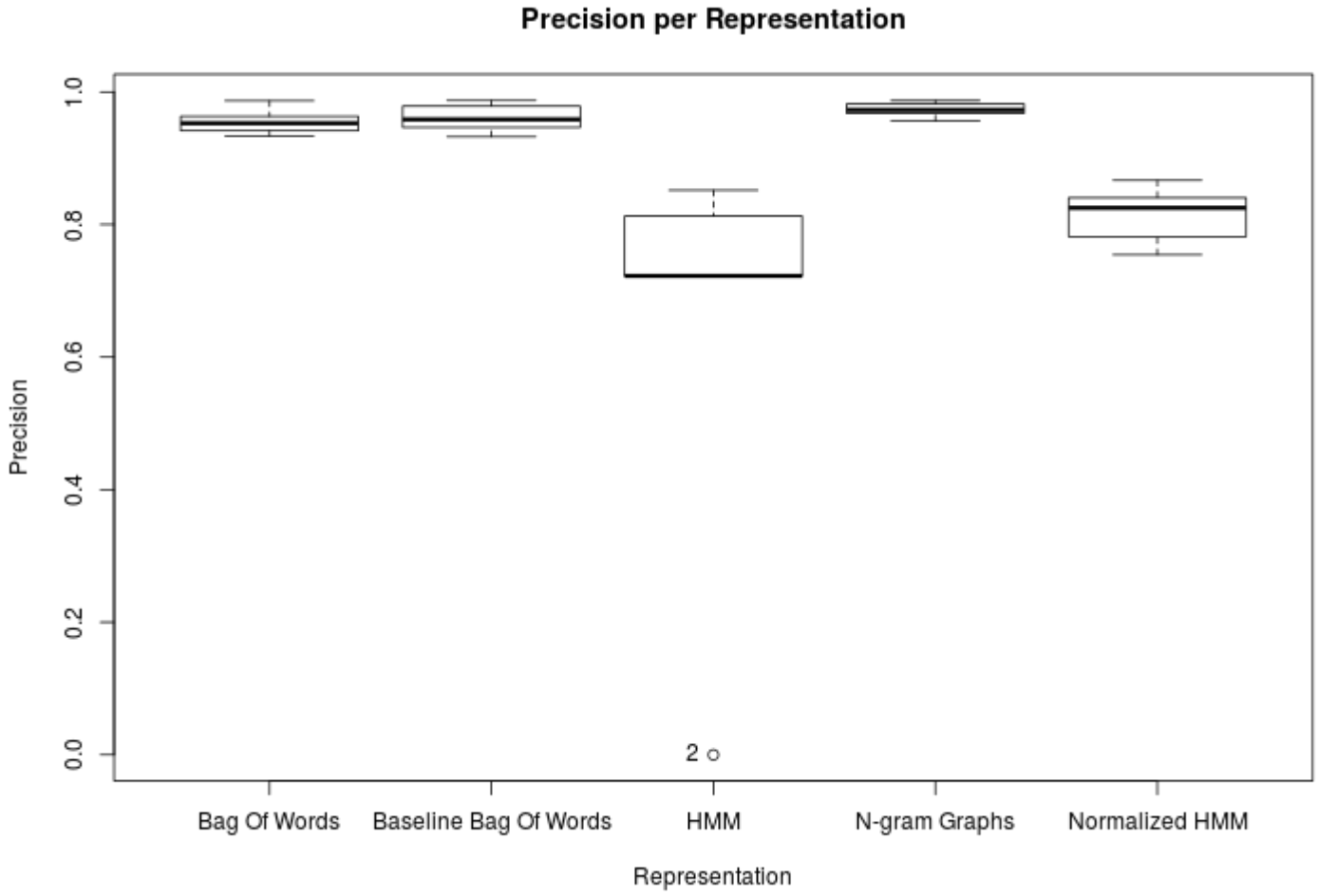

Figure 4.2: Precision measure per Representation.

The most successful models in terms of Precision are: the N-gram graphs and Baseline Bag-of-words models the mean Precision of which stands at 0.973 and 0.960 respectively. HMMs tend to be the least successful ones featuring a mean Precision of 0.769. Normalized HMMs appear to be more effective than plain ones with a mean precision at 0.814, while the Bag-of-words model featuring dimensionality reduction tends to be less precise than its Baseline model by a mean margin of 0.006.

Table 4.2: Mean Precision per Algorithm/Representation Pair (higher is better).

Representation	Classification Algorithm	Mean Precision
HMM	Naive Bayes	0.828
	Decision Trees	0.722
	kNN	0.803
	SVM	0.722
Normalized HMM	Naive Bayes	0.827
	Decision Trees	0.820
	kNN	0.840
	SVM	0.764
Bag-of-words	Naive Bayes	0.944
	Decision Trees	0.962
	kNN	0.970
	SVM	0.938
Baseline Bag-of-words	Naive Bayes	0.945
	Decision Trees	0.944
	kNN	0.971
	SVM	**0.982**
N-gram graphs	Naive Bayes	0.968
	Decision Trees	0.977
	kNN	0.980
	SVM	0.967

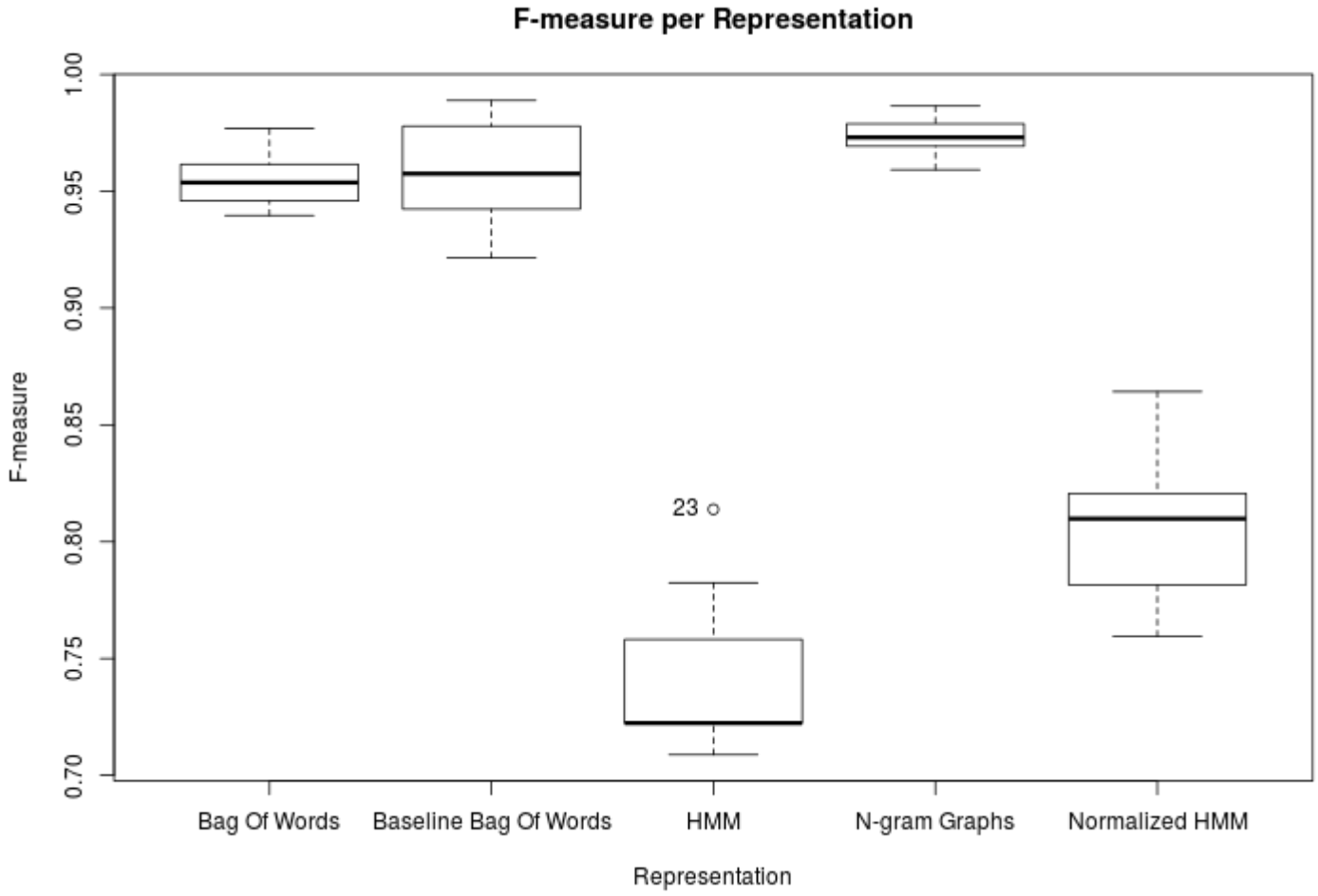

Figure 4.3: F-measure per Representation (NOTE: minimum y-axis value is 0.70).

The most successful models in terms of F-measure are: the Baseline Bag-of-words and N-gram graphs models the mean F-measure of which resides at 0.984 and 0.974 respectively. HMMs tend to be the least successful ones featuring a mean F-measure of 0.739. Normalized HMMs appear to be more effective than plain ones with a mean F-measure at 0.805, while the Bag-of-words model featuring dimensionality reduction tends to be less successful than its Baseline model by a mean margin of 0.004.

Table 4.3: Mean F-measure per Algorithm/Representation Pair (higher is better).

Representation	Classification Algorithm	Mean F-measure
HMM	Naive Bayes	0.747
	Decision Trees	0.722
	kNN	0.764
	SVM	0.722
Normalized HMM	Naive Bayes	0.824
	Decision Trees	0.824
	kNN	0.801
	SVM	0.770
Bag-of-words	Naive Bayes	0.947
	Decision Trees	0.963
	kNN	0.961
	SVM	0.945
Baseline Bag-of-words	Naive Bayes	0.951
	Decision Trees	0.932
	kNN	0.964
	SVM	**0.984**
N-gram graphs	Naive Bayes	0.969
	Decision Trees	0.972
	kNN	0.974
	SVM	0.971

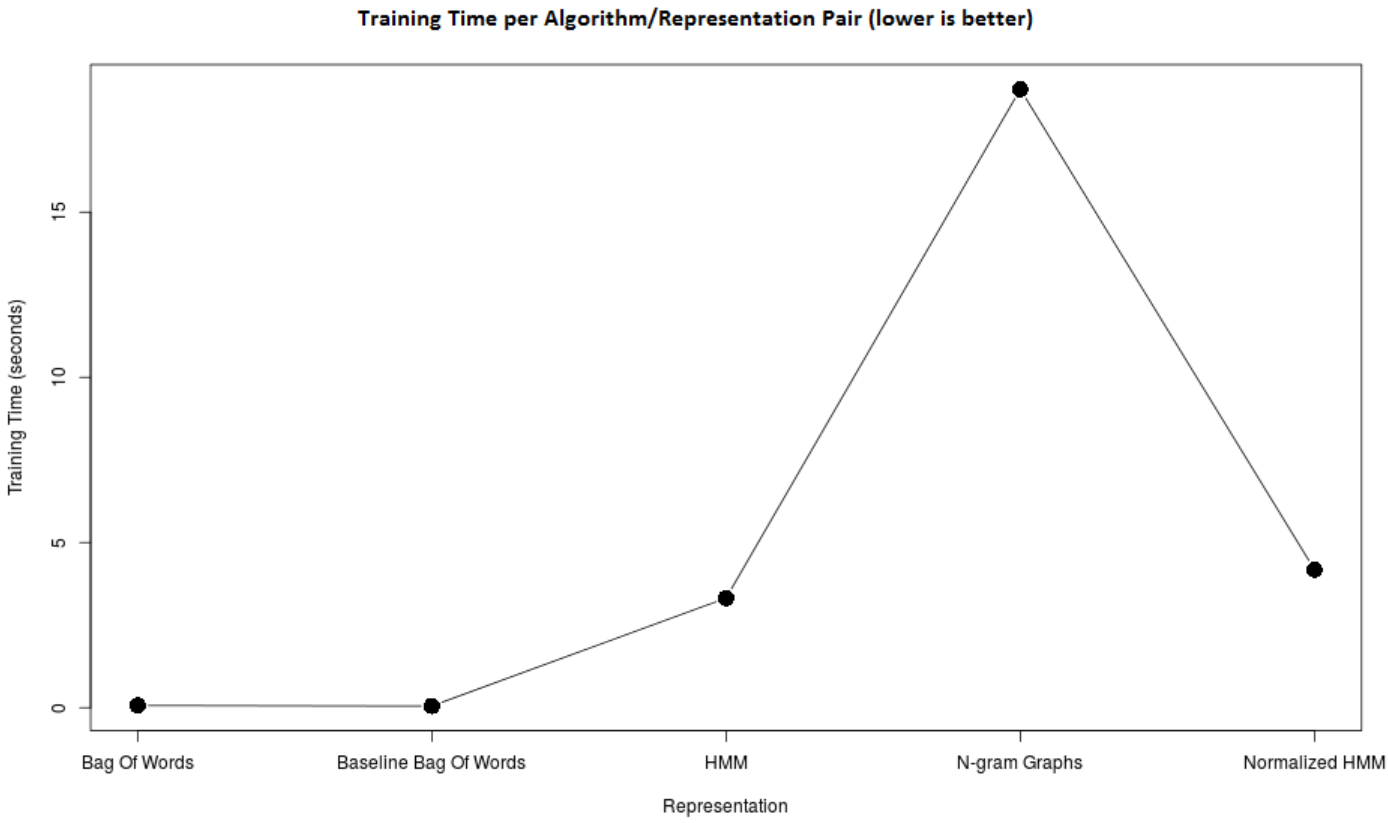

Figure 4.4: Training Time per Algorithm/Representation Pair.

In terms of training time both the Bag-of-words and Baseline Bag-of-words models are the fastest to train, as they take 0.0518 and 0.0674 seconds for the training phase to complete. The HMM models (i.e. plain HMM and Normalized HMM) take a quite longer to train, as they need 3.317 and 4.177 seconds to be trained. Finally, the most inefficient representation to train is the N-gram graphs model, using 18.718 seconds for the training phase.

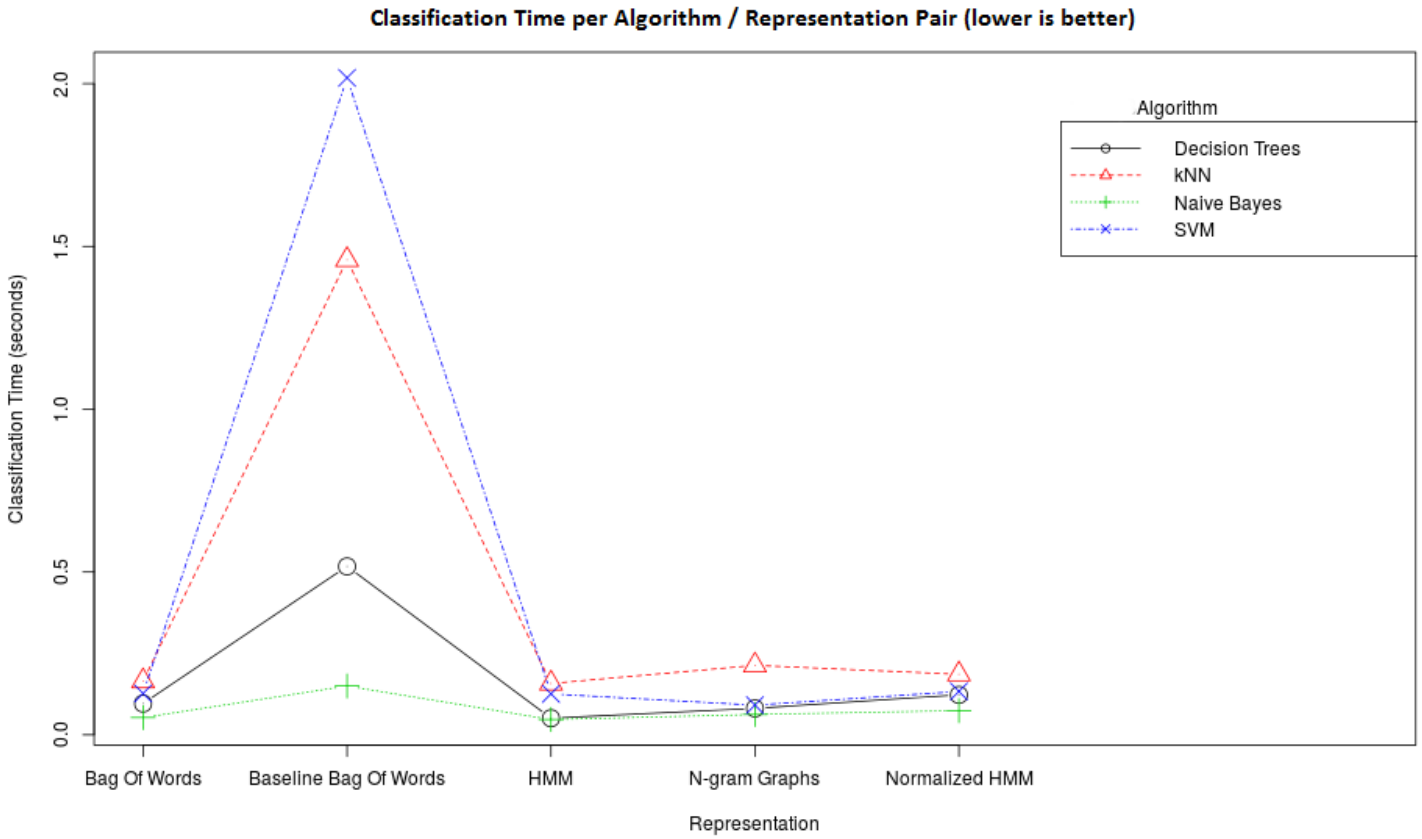

Figure 4.5: Classification Time per Algorithm/Representation Pair.

In terms of classification time all the models except for the Baseline Bag-of-words model take little time for the classification process to finish ranging from 0.046 to 0.16 seconds at most. The Baseline Bag-of-words model is the only one, the classification process of which takes multiple times to finish, in relation with the classification times of the rest of the models.

Table 4.4: Mean Classification Time per Algorithm/Representation Pair (lower is better).

Representation	**Classification Algorithm**	**Mean Classification Time** (secs)
HMM	Naive Bayes	**0.046**
	Decision Trees	0.050
	kNN	0.157
	SVM	0.125
Normalized HMM	Naive Bayes	0.074
	Decision Trees	0.122
	kNN	0.185
	SVM	0.133
Bag-of-words	Naive Bayes	0.052
	Decision Trees	0.095
	kNN	0.164
	SVM	0.127
Baseline Bag-of-words	Naive Bayes	0.149
	Decision Trees	0.517
	kNN	1.458
	SVM	2.017
N-gram graphs	Naive Bayes	0.062
	Decision Trees	0.095
	kNN	0.213
	SVM	0.091

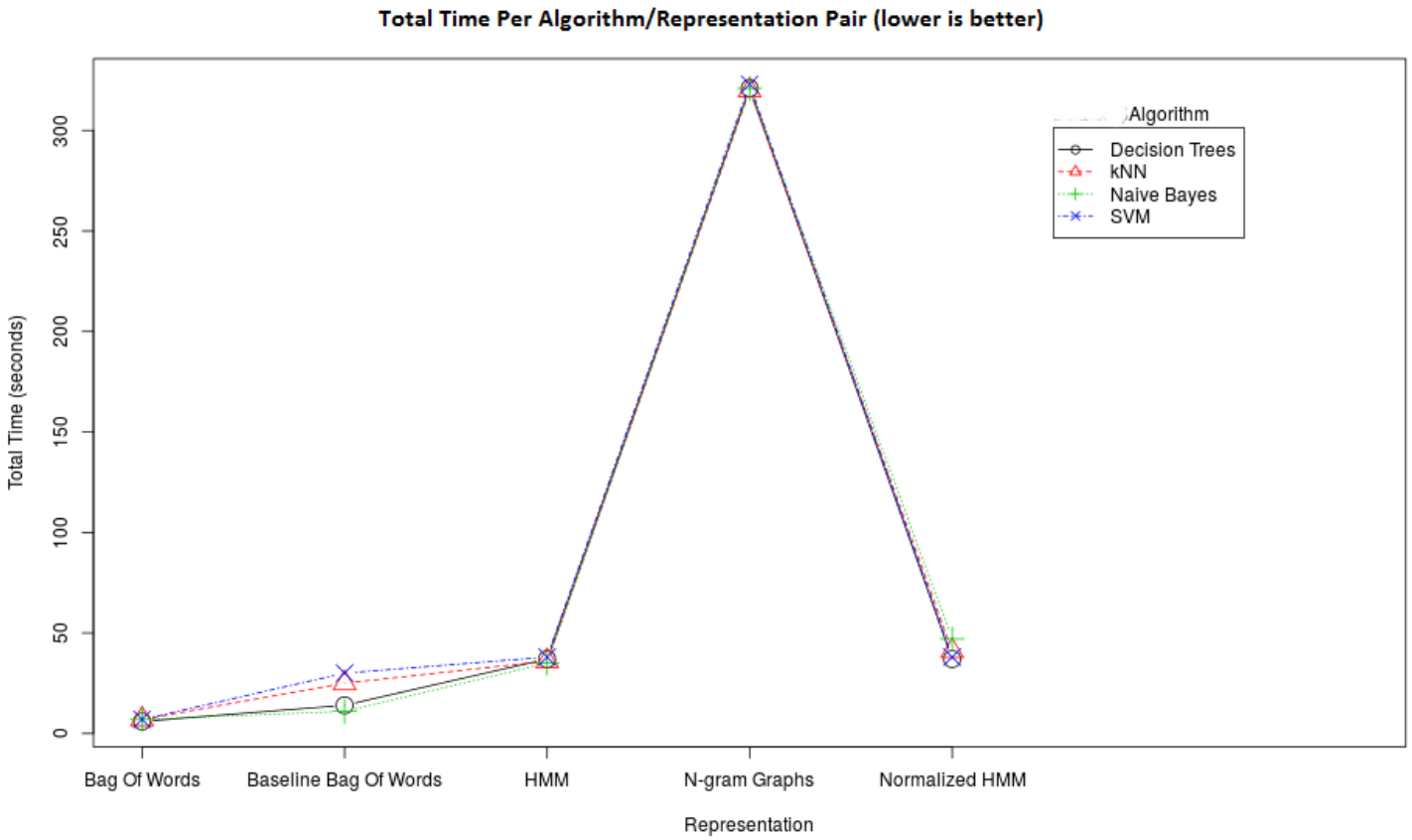

Figure 4.6: Total Time per Algorithm/Representation Pair.

The most efficient models concerning completion time (i.e. training time + classification time) are the Bag-of-words which take 6-7 seconds to finish the training and classification processes. The Baseline Bag-of-words follow with 11-30 seconds to completion, while the HMM and Normalized HMM ones take relatively equal time to finish, at 35-38 and 37-47 respectively. The most inefficient model, in terms of completion time, are the N-gram graphs, the completion time of which takes up to 320-323 seconds.

Table 4.5: Mean Total Time per Algorithm/Representation Pair (lower is better).

Representation	**Classification Algorithm**	**Mean Total Time** (secs)
HMM	Naive Bayes	35
	Decision Trees	37
	kNN	36
	SVM	38
Normalized HMM	Naive Bayes	47
	Decision Trees	37
	kNN	41
	SVM	38
Bag-of-words	Naive Bayes	7
	Decision Trees	**6**
	kNN	7
	SVM	7
Baseline Bag-of-words	Naive Bayes	11
	Decision Trees	14
	kNN	25
	SVM	30
N-gram graphs	Naive Bayes	321
	Decision Trees	321
	kNN	320
	SVM	323

5. DISCUSSION AND CONCLUSION

Here we are going to sum up our conclusions resulted from the experiments we've conducted so far.

First and foremost, a striking fact to note is that the dataset imbalance in our classes affected negatively the efficiency of our experiments. The class instances were 1033 for the NFR class and 3061 for the NBS class, therefore some models and especially the HMM suffered from the Most Common Class problem. However, all the other models, the Normalized HMM included, fared better against these problems and proved to tackle them efficiently.

Furthermore, our results indicated that the N-gram graphs and Bag-of-words models are better suited for our problem than HMM. This is better shown in **section 4.2** where comparisons between the different models are made, based on the Recall, Precision and F-measure metrics.

Another thing to note, is that, although the N-gram graphs model is more efficient than the HMM, the time reserved for its training phase overwhelms that of the other models. However, due to the training phase's unique heavy preprocessing schedule, the training time is reasonably lower, in comparison to the other models. Moreover, we should also include the fact that the Normalized version of the HMM model tended to be more efficient than the plain one, which is probably correlated with the normalization procedure that we used.

Finally, we can't ignore the fact that the Baseline Bag-of-words model has the longest classification time among our models. This is probably explained by the fact that the features extracted from this particular model were high-dimensional, which proved to constrain the efficiency of the classification algorithms used. This particular problem, though, doesn't appear in the dimensionality-reduced case of our standard Bag-of-words model, a fact which reinforces our intuition about the dimensionality problem being apparent in the first model.

Some of the next steps towards increasing our understanding of global chromatin structure from a Machine Learning perspective will be to introduce new methods which could possibly complement the already existent ones. As the information era moves forward, large computational power and vast datasets have become the norm in research studies, facilitating the acquisition of better results more efficiently. Therefore, a research direction

could be to apply the current methods to larger and more complex datasets, such as the *H.sapiens* genome, to verify that similar results can be found in larger eukaryotic systems and produce more interesting results.

Furthermore, another promising area of research could be to optimise the feature extraction so far used between different representations, or even introduce new representations for a wider variety of possible features. Better features can ultimately lead to better classification results which can capture more accurately the factors that distinguish genomic sequences between Nucleosome Free Regions and Nucleosome Binding Sites.

Last but not least, an important decision to make is whether the above results conserning genomic sequences could be enhanced by analyzing the secondary structure of DNA, which gives us a wider view of the interactions between bases, i.e., which parts of strands are bound to each other.

ABBREVIATIONS - ACRONYMS

ABBREVIATION	FULL COMMENTARY
NBS	Nucleosome Binding Site
NFR	Nucleosome Free Region
HMM	Hidden Markov Model
HMM	Hidden Markov Model
NGG	N-gram Graph
BOW	Bag-of-words
DNA	Deoxyribonucleic acid
MMTV	Mouse Mammary Tumor Virus
TSS	Transcription Start Site
ML	Machine Learning
EM	Estimation Maximization
kNN	k-Nearest Neighbors
SVM	Support Vector Machine
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives

BIBLIOGRAPHY

[1] Matthew G. Guenther, Stuart S. Levine, Laurie A. Boyer, Rudolf Jaenisch, and Richard A. Young. A chromatin landmark and transcription initiation at most promoters in human cells. 130(1):77–88.

[2] Matthew L. Eaton, Kyriaki Galani, Sukhyun Kang, Stephen P. Bell, and David M. MacAlpine. Conserved nucleosome positioning defines replication origins. 24(8):748–753.

[3] Heather E. Peckham, Robert E. Thurman, Yutao Fu, John A. Stamatoyannopoulos, William Stafford Noble, Kevin Struhl, and Zhiping Weng. Nucleosome positioning signals in genomic DNA. 17(8):1170–1177.

[4] Guillermo P. Vicent, A. Silvina Nacht, Corey L. Smith, Craig L. Peterson, Stefan Dimitrov, and Miguel Beato. DNA instructed displacement of histones h2a and h2b at an inducible promoter. 16(3):439–452.

[5] Guo-Cheng Yuan, Yuen-Jong Liu, Michael F. Dion, Michael D. Slack, Lani F. Wu, Steven J. Altschuler, and Oliver J. Rando. Genome-scale identification of nucleosome positions in s. cerevisiae. 309(5734):626–630.

[6] William Lee, Desiree Tillo, Nicolas Bray, Randall H. Morse, Ronald W. Davis, Timothy R. Hughes, and Corey Nislow. A high-resolution atlas of nucleosome occupancy in yeast. 39(10):1235–1244.

[7] Sushma Shivaswamy, Akshay Bhinge, Yongjun Zhao, Steven Jones, Martin Hirst, and Vishwanath R. Iyer. Dynamic remodeling of individual nucleosomes across a eukaryotic genome in response to transcriptional perturbation. 6(3):e65.

[8] Arnold Stein, Taichi E. Takasuka, and Clayton K. Collings. Are nucleosome positions in vivo primarily determined by histone–DNA sequence preferences? 38(3):709–719.

[9] Micaela Caserta, Eleonora Agricola, Mark Churcher, Edwige Hiriart, Loredana Verdone, Ernesto Di Mauro, and Andrew Travers. A translational signature for nucleosome positioning in vivo. *Nucleic Acids Research*, 2009.

[10] Noam Kaplan, Irene K. Moore, Yvonne Fondufe-Mittendorf, Andrea J. Gossett, Desiree Tillo, Yair Field, Emily M. LeProust, Timothy R. Hughes, Jason D. Lieb, Jonathan Widom, and Eran Segal. The DNA-encoded nucleosome organization of a eukaryotic genome. 458(7236):362–366.

[11] Eran Segal, Yvonne Fondufe-Mittendorf, Lingyi Chen, AnnChristine Thåstrom, Yair Field, Irene K. Moore, Ji-Ping Z. Wang, and Jonathan Widom. A genomic code for nucleosome positioning. 442(7104):772–778.

[12] Ryu Ogawa, Noriyuki Kitagawa, Hiroki Ashida, Rintaro Saito, and Masaru Tomita. Computational prediction of nucleosome positioning by calculating the relative fragment frequency index of nucleosomal sequences. 584(8):1498–1502.

[13] Thomas Bettecken and Edward N. Trifonov. Repertoires of the nucleosome-positioning dinucleotides. 4(11):e7654.

[14] Roger D. Kornberg and Lubert Stryer. Statistical distributions of nucleosomes: non-random locations by a stochastic mechanism. 16(14):6677–6690.

[15] Travis N. Mavrich, Ilya P. Ioshikhes, Bryan J. Venters, Cizhong Jiang, Lynn P. Tomsho, Ji Qi, Stephan C. Schuster, Istvan Albert, and B. Franklin Pugh. A barrier nucleosome model for statistical positioning of nucleosomes throughout the yeast genome. 18(7):1073–1083.

[16] Sheila M. Reynolds, Jeff A. Bilmes, and William Stafford Noble. Learning a weighted sequence model of the nucleosome core and linker yields more accurate predictions in ¡italic¿saccharomyces cerevisiae¡/italic¿ and ¡italic¿homo sapiens¡/italic¿. *PLoS Comput Biol*, 6(7):1–17, 07 2010.

[17] Christoforos Nikolaou, Sonja Althammer, Miguel Beato, and Roderic Guigo. Structural constraints revealed in consistent nucleosome positions in the genome of s. cerevisiae. 3(1):20.

[18] Ryoiti Kiyama and Edward N Trifonov. What positions nucleosomes? – a model. 523(1):7–11.

[19] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., 1 edition.

[20] Christopher D. Manning and Hinrich Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press.

[21] Panagiota Antonakaki, Dimitrios Kosmopoulos, and Stavros J. Perantonis. Detecting abnormal human behaviour using multiple cameras. *Signal Process.*, 89(9):1723–1738, September 2009.

[22] George Giannakopoulos, Vangelis Karkaletsis, George Vouros, and Panagiotis Stamatopoulos. Summarization system evaluation revisited: N-gram graphs. 5(3):5:1–5:39.

[23] F. Aisopos, D. Tzannetos, J. Violos, and T. Varvarigou. Using n-gram graphs for sentiment analysis: An extended study on twitter. In *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 44–51, March 2016.

[24] George Giannakopoulos. *Automatic Summarization from Multiple Documents*. PhD thesis, University of the Aegean, 2009.

[25] Sreerama K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. 2(4):345–389.

[26] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas. Machine learning: a review of classification and combining techniques. 26(3):159–190.

[27] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition.

[28] Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, 37(6):1554–1563, 12 1966.

[29] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.