Automated Extraction of Associations between Methylated Genes and Diseases from Biomedical Literature

Thesis by

Arwa Bin Res

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science

King Abdullah University of Science and Technology

Thuwal, Kingdom of Saudi Arabia

December, 2012
The thesis of Arwa Bin Res is approved by the examination committee.

Committee Chairperson: Vladimir Bajic
Committee Member: Mikhail Moshkov
Committee Member: Xiangliang Zhang
ABSTRACT

Automated Extraction of Associations between Methylated Genes and Diseases from Biomedical Literature

Arwa Bin Res

Associations between methylated genes and diseases have been investigated in several studies, and it is critical to have such information available for better understanding of diseases and clinical decisions. However, such information is scattered in a large number of electronic publications and it is difficult to manually search for it. Therefore, the goal of the project is to develop a machine learning model that can efficiently extract such information. Twelve machine learning algorithms were applied and compared in application to this problem based on three approaches that involve: document-term frequency matrices, position weight matrices, and a hybrid approach that uses the combination of the previous two. The best results we obtained by the hybrid approach with a random forest model that, in a 10-fold cross-validation, achieved F-score and accuracy of nearly 85% and 84%, respectively. On a completely separate testing set, F-score and accuracy of 89% and 88%, respectively, were obtained. Based on this model, we developed a tool that automates extraction of associations between methylated genes and diseases from electronic text. Our study contributed an efficient method for extracting specific types of associations from free text and the methodology developed here can be extended to other similar association extraction problems.
ACKNOWLEDGEMENT

This project was extremely worthwhile and enjoyable. I appreciate that research is a cautious, thoughtful process that can satisfy my curiosity for different topics. Doing research is like learning a new craft: it requires time, effort and dedication.

I am thankful to my supervisor, Prof. Vladimir Bajic, for his support, supervision and encouragement during the 14 months I spent working part-time on my thesis. I consider myself lucky to have got an opportunity to be supervised by a great supervisor like him.

Also, I would like to thank the committee members Prof. Mikhail Moshkov and Prof. Xiangliang Zhang. I want to thank Dr. Hicham Mansour and Dr. Roberto Incitti for helping me with the datasets I used. Also, I want to thank Dr. Aleksandar Radovanovic for helping me with the dictionaries of diseases and genes.

Nobody has been more supportive to me during this journey than the members of my family. I would like to thank my parents who believed in me in whatever I pursue.
# TABLE OF CONTENTS

Examination Committee Approvals Form ......................................................... 2
Copyright............................................................................................................ 3
ABSTRACT.......................................................................................................... 4
ACKNOWLEDGEMENT........................................................................................ 5
LIST OF ABBREVIATIONS ................................................................................ 8
LIST OF FIGURES .............................................................................................. 9
LIST OF TABLES .............................................................................................. 10

Chapter 1 Introduction .................................................................................... 12
  1.1 Research Focus, Goal and Objectives ....................................................... 14
  1.2 Background ............................................................................................. 15
    1.2.1 Text Mining ..................................................................................... 15
    1.2.2 Text Mining for Biomedical Literature .......................................... 16
    1.2.3 Epigenetics ...................................................................................... 18
    1.2.4 DNA Methylation .......................................................................... 19
    1.2.5 DNA Methylation and Diseases ..................................................... 20

Chapter 2 Literature Review .......................................................................... 23
  2.1 Associations extraction tools from biomedical literature ....................... 23
  2.2 Tools that accept user-provided text ....................................................... 25
  2.3 Methylation databases ............................................................................ 26
  2.4 Emerging issues and the need for further research ............................... 30

Chapter 3 Methodology ............................................................................... 32
  3.1 Problem Formulation ............................................................................. 32
  3.2 Dataset .................................................................................................... 35
  3.3 Data Preprocessing .................................................................................. 35
    3.3.1 Sentence Boundary Determination ................................................. 35
    3.3.2 Tokenization .................................................................................. 36
    3.3.3 Named Entity Recognition ............................................................ 37
    3.3.4 Stop-words Elimination ................................................................. 37
    3.3.5 Stemming or Lemmatization ......................................................... 38
    3.3.6 Keywords Selection ...................................................................... 38
  3.4 Structured Data Representation .............................................................. 39
    3.4.1 Document-Term Frequency Matrix (DTFM) ............................... 40
    3.4.2 Position Weight Matrices (PWMs) ............................................... 41
    3.4.3 PWMs Generation from Biomedical Literature ......................... 42
    3.4.4 Computing the Scores from PWMs ............................................. 45
    3.4.5 Feature Generation ...................................................................... 46
Chapter 4 Experiments and Results

4.1 Performance Metrics .......................................................... 56
4.2 PWMs Approach .................................................................. 57
4.3 DTFM Approach .................................................................. 59
  4.3.1 Rule Generation Algorithms ............................................ 59
  4.3.2 Decision Trees Algorithms .............................................. 61
  4.3.3 SVM ........................................................................... 63
  4.3.4 KNN Algorithm .............................................................. 65
  4.3.5 Naïve Bayes Algorithm .................................................... 66
4.4 Hybrid Approach .................................................................. 67
4.5 Testing Results .................................................................... 68
4.6 Summary of Results .............................................................. 68

Chapter 5 Discussion ................................................................. 73
5.1 Comparison between PWMs, DTFM and Hybrid Approaches ....... 73
5.2 Analysis of Algorithms’ Results ............................................. 75
  5.2.1 Rule Generation Algorithms .......................................... 76
  5.2.2 Decision Trees Algorithms ............................................ 76
  5.2.3 SVM .......................................................................... 77
  5.2.4 KNN ........................................................................... 77
  5.2.5 Naïve Bayes Algorithm .................................................. 78
5.3 Problems and Limitations ....................................................... 78

Chapter 6 Conclusion ................................................................. 80
6.1 Summary of Findings ........................................................... 80
6.2 Summary of Contributions .................................................. 81
6.3 Recommendations .............................................................. 83

BIBLIOGRAPHY ....................................................................... 86
## LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>accuracy</td>
</tr>
<tr>
<td>BOW</td>
<td>Boundary Of Words</td>
</tr>
<tr>
<td>CBA</td>
<td>Classification Based on Associations</td>
</tr>
<tr>
<td>CMAR</td>
<td>Classification based on Multiple Association Rules</td>
</tr>
<tr>
<td>CPAR</td>
<td>Classification based on Predictive Association Rules</td>
</tr>
<tr>
<td>DTFAM</td>
<td>Dragon TF Association Miner</td>
</tr>
<tr>
<td>DTFM</td>
<td>Document-Term Frequency Matrix</td>
</tr>
<tr>
<td>DPBE</td>
<td>Dragon Plant Biology Explorer</td>
</tr>
<tr>
<td>EOS</td>
<td>End Of Sentences</td>
</tr>
<tr>
<td>FOIL</td>
<td>First Order Induction Logic</td>
</tr>
<tr>
<td>FN</td>
<td>false negative</td>
</tr>
<tr>
<td>FP</td>
<td>false positive</td>
</tr>
<tr>
<td>IG</td>
<td>information gain</td>
</tr>
<tr>
<td>KNN</td>
<td>K-nearest neighbor</td>
</tr>
<tr>
<td>MCS</td>
<td>Minimum Chi-Square</td>
</tr>
<tr>
<td>PR</td>
<td>precision</td>
</tr>
<tr>
<td>PRM</td>
<td>Predictive Rule Mining</td>
</tr>
<tr>
<td>PWM</td>
<td>position weight matrix</td>
</tr>
<tr>
<td>RE</td>
<td>recall</td>
</tr>
<tr>
<td>SP</td>
<td>Specificity</td>
</tr>
<tr>
<td>SVM</td>
<td>support vector machines</td>
</tr>
<tr>
<td>TF-IDF</td>
<td>term-frequency inverse-document-frequency</td>
</tr>
<tr>
<td>TFPC</td>
<td>Total From Partial Classification</td>
</tr>
<tr>
<td>TN</td>
<td>True Negative</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive</td>
</tr>
<tr>
<td>WCS</td>
<td>Weighted Chi-Square</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1-1: DNA methylation and Histone Modifications (from [15]) ........................................ 19
Figure 1-2: Methylation of Tumor Suppressor Genes (from [21]) .................................................. 21
Figure 2-1: DPBE interface to submit abstracts (from http://research.i2r.a-star.edu.sg/DRAGON/ME2/) ................................................................. 26
Figure 2-2: DiseaseMeth Search Engine (from http://bioinfo.hrbmu.edu.cn/diseasemeth) ................... 28
Figure 2-3: MethyCancer Search Engine (from http://methycancer.genomics.org.cn) .... 29
Figure 2-4: MethDB Search Engine (from http://www.methdb.net) ................................................. 29
Figure 3-1: DTFM representation ........................................................................................................ 41
Figure 3-2: PWMs ............................................................................................................................... 43
Figure 3-3: PWMs Generation Example ................................................................................................ 44
Figure 3-4: Example for Computing the Score .................................................................................... 46
Figure 3-5: Illustration of Features Generated by PWMs .................................................................. 47
Figure 3-6: System Architecture ........................................................................................................... 55
Figure 4-1: Summary of 10-fold cross-validation accuracy using PWMs approach on set C .......................................................... 69
Figure 4-2: Best 10-fold cross-validation accuracy of rule generation when applied to the entire set C and sentences with one pattern on set C ..................................................... 69
Figure 4-3: 10-fold cross-validation accuracy of algorithms using DTFM approach on set C .......................................................... 70
Figure 4-4: 10-fold cross-validation accuracy using DTFM approach on sentences with one pattern on set C ........................................................................................................ 70
Figure 4-5: 10-fold cross-validation accuracy of Naive Bayes on set C and sentences with one pattern on set C ........................................................................................................ 71
Figure 4-6: Best 10-fold cross-validation accuracy of all algorithms using DTFM approach on set C ................................................................................................. 71
Figure 4-7: Comparison of 10-fold cross-validation accuracy between DTFM and PWMs approaches on set C ................................................................................................. 72
Figure 4-8: Comparison between 10-fold cross-validation accuracy of Hybrid, PWM and DTFM approaches on set C ................................................................................................. 72
LIST OF TABLES

Table 2-1: Summary of association extraction tools (ACC = accuracy, RE = recall and PR = precision) ................................................................. 24

Table 3-1: Illustration of pattern order .............................................................. 33

Table 3-2: Sample DNA Sequences ................................................................. 41

Table 3-3: Position Frequency Matrix ............................................................. 42

Table 3-4: Position Weight Matrix ................................................................. 42

Table 3-5: Features of two patterns in the same sentence .................................. 47

Table 4-1: Values of parameters set for different algorithms ............................. 57

Table 4-2: 10-fold cross-validation results on set C using sentences with one pattern only ................................................................. 58

Table 4-3: 10-fold cross-validation results on set C using sentences with multiple patterns only ........................................................................................................ 58

Table 4-4: 10-fold cross-validation results on set C ............................................ 59

Table 4-5: 10-fold cross-validation accuracy after using different gain values on set C . 60

Table 4-6: 10-fold cross-validation accuracy after using different gain values on sentences with one pattern from set C ........................................................................................................ 60

Table 4-7: 10-fold cross-validation accuracy of algorithms on set C ..................... 60

Table 4-8: 10-fold cross-validation accuracy of algorithms on one pattern sentences on set C ........................................................................................................ 60

Table 4-9: 10-fold cross-validation accuracy of algorithms after applying them on set C61

Table 4-10: Best 10-fold cross-validation accuracy of decision trees on set C .......... 62

Table 4-11: 10-fold cross-validation accuracy of algorithms after applying them on sentences with one pattern on set C ........................................................................................................ 62

Table 4-12: The best 10-fold cross-validation accuracy of decision trees on sentences with one pattern on set C ........................................................................................................ 63

Table 4-13: 10-fold cross-validation accuracy of SVM after applying it on set C ....... 63
Table 4-14: The best 10-fold cross-validation accuracy of SVM after applying it on set C
........................................................................................................................................... 64

Table 4-15: 10-fold cross-validation accuracy of SVM after applying it on sentences with
one pattern only on set C ........................................................................................................... 64

Table 4-16: Best 10-fold cross-validation accuracy of SVM after applying it on sentences
with one pattern on set C ........................................................................................................... 64

Table 4-17: 10-fold cross-validation accuracy of KNN on set C ............................................. 65

Table 4-18: 10-fold cross-validation accuracy of KNN after using different number of
neighbors on set C .................................................................................................................. 65

Table 4-19: 10-fold cross-validation accuracy of KNN after applying it on sentences with
one pattern on set C .................................................................................................................. 66

Table 4-20: 10-fold cross-validation accuracy of KNN by using different number of
neighbors on sentences with one pattern on set C ............................................................... 66

Table 4-21: 10-fold cross-validation accuracy of Naïve Bayes after applying it on set C67

Table 4-22: 10-fold cross-validation accuracy of Naive Bayes after applying it on
sentences with one pattern on set C .......................................................................................... 67

Table 4-23: 10-fold cross-validation accuracy of different algorithms using the hybrid
approach on set C ....................................................................................................................... 68

Table 4-24: Performance of the best algorithm on set T .......................................................... 68
Chapter 1 Introduction

The aim of this chapter is to introduce the background of the problem we studied, discuss the research focus and identify overall research goals and objectives. This chapter provides the reader with background information on text mining, including an illustration of how text mining for biomedical literature poses different challenges that cannot be found in other data mining problems. Also, the chapter provides background information on epigenetic modifications and DNA methylation and how they are linked with disease development and prognosis.

Most of the scientific knowledge we obtained through research is contained in scientific literature [1] which uses natural language. This allows to communicate and share information within the scientific community. Modern biology evolves continuously and biological research topics change quickly [2]. Also, a large amount of biological data can be produced rapidly using advanced experimental techniques [3]. Information of interest can be gathered from scientific literature in order to analyze generated data and derive new conclusions. Eventually, these results are published in peer-reviewed articles, hence enlarging the volume of the current literature. An increasing number of electronic documents such as reports, patents, and scientific publications are accessible through the World Wide Web, and contain most of biological knowledge [2].

The information from natural language within electronic documents is understandable by humans only. The practical problem is that for any topic that is not extremely narrow, the volume of documents that contain information of interest is too large for a single person to analyze it manually. On the other hand, current computers can process the huge size of
data, but cannot interpret its meaning. Careful reading, interpretation and information annotation is necessary to gather and utilize the domain specific knowledge [1] from scientific literature. As a result, the life science community has witnessed development of many manually curated databases [2] related to specific topics. However, such databases can cover a small portion of information found in biological literature, and cannot absorb fully the richness of currently available information. Even though there is a tendency to enlarge expert-curated databases, the manual annotation process is slow and is associated with an expensive manual labor. Keeping such databases up-to-date is a very challenging task, because it requires human-curators to process an ever increasing number of new publications [2].

It is not surprising that there is a huge emphasis on developing knowledge discovery techniques that can discover new knowledge, identify, manage and utilize this knowledge, and present it to end-users in a concise form [4]. Knowledge acquired from previous studies can be used to design experimental settings, to choose topics to be studied or to identify biological questions to be answered [2]. Therefore, there is a need for innovative algorithms to extract valuable information from free text, and to cope with the increasing volume of biological literature [1, 3]. Computer programs are now able to extract to an extent meaningful knowledge from research articles by means of text mining and natural language processing techniques [5]. These methodologies when focused on finding meaningful associations between biomedical entities can lead to new discoveries and thus can change the way people view and read scientific literature.
1.1 Research Focus, Goal and Objectives

Associations between methylated genes and diseases have been investigated by several recent studies [6-10], and it is critical to keep such information updated for better understanding of diseases and clinical decisions. However, information on genes methylated in different diseases is scattered in a large number of electronic publications, which makes it difficult to search for it manually. For this reason, the goal of the study is to develop machine learning models that can extract such information. This task involves specifying certain entity types (i.e., genes and diseases) and the relation between them (i.e., methylation) that may exist in free text. Therefore, the objectives of this research are to:

- Review scientific literature to explore new advances related to automated extraction of associations between methylated genes and diseases, and more generally associations between two entities.
- Compile datasets to be used in this study.
- Develop a method to generate the necessary features for the association extraction problem.
- Based on comparison of various machine learning models, determine the best performing one.
- Based on the best model type develop a system to automate the process of associations extraction between methylated genes and diseases.
1.2 Background

This section provides broad information related to this study. Firstly, section 1.2.1 Text Mining defines text mining and illustrates how text requires a special form of data representation. Then section 1.2.2 Text Mining for Biomedical Literature conveys inherit challenges in mining biomedical literature. Next, section 1.2.3 Epigenetics provides short background on epigenetics changes. After that, section 1.2.4 DNA Methylation discusses methylation as one type of epigenetic changes. Finally, section 1.2.5 DNA Methylation and Diseases shows how epigenetic modifications, specifically DNA methylation, are linked to diseases generally and cancer specifically.

1.2.1 Text Mining

Text mining, one of the fields of data mining, is “the discovery and extraction of interesting, non-trivial knowledge from free unstructured text” [11]. Among other things, text mining includes text classification, text clustering, information extraction and information retrieval [11]. Text mining can identify relevant named entities (such as drugs, diseases, genes, etc.) [12]. Also, it can be used to recognize associations and links between these entities such as protein-protein interactions, or genes and diseases associations [5, 12].

The nature of data used in text mining is different than the nature of data used in data mining. Free text we find in documents is in unstructured format. Therefore, an additional step is necessary in order to transform text in a suitable format so that data mining applications can process it. This approach can be useful to solve some text mining tasks such as text classification or text clustering. However, bigger problems, such as
answering specific questions, require more understanding of the language, which can be handled by natural language processing [13].

Natural language processing, one of the fields of text mining, aims to extract certain information from text (e.g., hypothetical statements, assertions, facts, etc.) [12]. This requires use of grammatical structures such as identifying noun phrases vs. verb phrases. It uses part-of-speech tagging concept, so as to determine, for example, if a word is a noun or a verb. It utilizes various resources such as thesaurus of abbreviations, ontologies, or grammar rules [11].

1.2.2 Text Mining for Biomedical Literature

Large amounts of information can be extracted by using natural language processing approaches. However, compared to general articles, biomedical text is considered a special challenge for text miners because of its specialized language, diversity, and ambiguity. Biological text has unique characteristics that makes it hard or even impractical for general-purpose text mining tools to process [1].

Life sciences literature is based on using concepts such as names of genes or proteins. It is a crucial step to identify the genes or proteins in the text in order to extract information about them. However, genes and proteins can be mentioned in text in various ways because of the complexity and nature of gene names, and this makes gene or protein names extraction a complex process [3]. For example, a protein can be mentioned by using its full name (*Gata Binding Protein2*), its abbreviation (*GBP2*), or by using typographical variations (such as *GBP-2* or *GBP 2*). This level of word variation can hamper retrieval tasks, and it makes parsing and tokenizing challenging processes [2, 4].
Any parser for biomedical literature must operate on one character at a time in order to identify certain characters with special meanings. Examples are using ‘.’ to indicate the end of a sentence or in a decimal number (e.g., 2.01) or making it part of a name (e.g. E. coli), using ‘/’ to connect several words (such as ‘waf/cip-1’) or using whitespaces to express names of genes as mentioned previously (GBP 2 vs. GBP-2) [2].

Biological research uses a specialized language which is usually ambiguous [1]. Ambiguity occurs when a new abbreviation is found to be similar to an abbreviation for another term, or describing new ideas by using existing terms [2]. Gene names may correspond to some words that have different meaning, and selecting the right meaning requires understanding of the context first [2, 3]. For example, if we consider these two sentences ‘The Drosophila peanut gene is required for cytokinesis’ (PMID 8181057) and ‘Peanut (Arachis hypogaea) forms root nodules in a unique process.’ (PMID 1825023), Peanut in the first sentence is a name of a fly gene, whereas in the second sentence it is a name of a plant [2]. In fact, gene names have 14% ambiguity, which is relatively large compared with 0.57% ambiguity in general English language [1, 2].

Biomedical terminology is one of the main challenges in text mining [4]. One of the main difficulties when it comes to dealing with terminology is persistent increase of new abbreviations or terms [2]. For example, PTA can have one of the following meanings: post-traumatic amnesia, purified terephthalic acid, percutaneous tumor ablation, pre-T cell receptor alpha, and polyclonal T-cell activators (based on the Acromine tool [14]). In fact, on average, there is one new abbreviation introduced in every 5 – 10 PubMed abstracts, and it is estimated that there is a total of 800,000 different abbreviations in
biomedical literature [1]. This is due to the fact that biology interacts with several disciplines, such as medicine and chemistry, and this interdisciplinary nature of biology must be considered when developing text mining tools for biological text [4].

1.2.3 Epigenetics

Each cell in human body functions based on activity of a large number of genes. However, in each cell type only some of them are active [15]. The genes are segments of the DNA molecule and the activation of genes is not isolated from surrounding environment in cells. Gene activity is mainly, but not only, controlled by the transcription factor proteins that interact with DNA and control gene activation [16].

Epigenetic modifications refer to “the extra layers of instructions that influence gene activity without altering the DNA sequence” [17]. Epigenetics is the study of these long-term, persistent, and heritable modifications in gene regulation without changing the underlying DNA sequence [16, 18-20]. Epigenetic modifications are important, because they control the type and features of each cell in the human body even though all cells share the same genome [19]. Epigenetic changes are not rare. Even though it seems that epigenetic changes occur randomly, there are several environmental factors that cause epigenetic changes. Examples are environmental pollutions, and even our dietary habits. Epigenetic gene regulation occurs more frequently as a result of our interactions with the environment [16]. Activation of each gene is controlled mainly through its promoter region. Most epigenetic changes take place at the promoter. It is a long-term, long-lasting and stable gene regulation, and it can be passed on to the next generation [16].
1.2.4 DNA Methylation

Different epigenetic mechanisms have been reported including, for example, phosphorylation, acetylation or methylation [8]. DNA methylation is one of the major and most known epigenetic modifications [7, 9, 21]. In this process (see Figure 1-1), a methyl group is attached to the DNA, which causes reduction in gene expression [21].

![The two main components of the epigenetic code](image)

Figure 1-1: DNA methylation and Histone Modifications (from [15])
DNA methylation does not change the DNA sequence. Rather, it changes the activation of the gene including altering gene expression, or silencing the transcription of the gene [9]. DNA methylation can be used as a protection mechanism against including foreign DNA or to reduce resistance from host DNA to foreign DNA [8]. Patterns of DNA methylation (like other epigenetic modifications) can be acquired during life or inherited [21].

1.2.5 DNA Methylation and Diseases

Defects in DNA methylation can cause human diseases, and disrupted epigenetic modifications are involved for example in cancers [8, 9]. Normally, promoter region of some tumor suppressor genes are un-methylated in healthy individuals but found to experience dense hyper-methylation in cancer patients [7]. Such a process (see Figure 1-2) is one of the most common mechanisms that results in silencing of expression of tumor suppressor genes or in reduction of their activity [7, 10, 21]. Therefore, in early tumor development, aberrant DNA methylation of tumor suppressor genes is found to be an indicator for cancers in humans and a mark for presence of a certain malignancy [7, 9]. We will be able to have an important prevention mechanism against cancers if we could control the methylation of these genes [21].

Aberrant DNA methylation is found to be related to tumor prognosis, tumor stage, and tumor grade [10]. Furthermore, it can be used to determine the tissue of origin for a specific cancer. Also, gene deregulation by DNA methylation in cancer can be a predictive marker to provide relevant information about patients’ response to chemotherapy treatment [7]. Therefore, methylation of tumor suppressor genes can be a
useful utility for diagnostic purposes to detect cancer or predict treatment outcomes [10]. In addition to cancer, aberrant DNA methylation is involved in several diseases including neurodevelopmental disorders, autoimmune diseases, and aging [9]. Also, modifications of DNA by methylation can cause cardiovascular diseases [7].

![Methylation of Tumor Suppressor Genes](image)

Figure 1-2: Methylation of Tumor Suppressor Genes (from [21])

Beyond that, Albright hereditary osteodystrophy, Russel-Silver syndrome, Prader-Willi/Angelman syndromes, and Beckwith-Wiedemann syndrome are good examples of DNA methylation dependent diseases [7]. Detection of DNA methylation can be a cost-effective, sensitive and accurate utility to detect tumors. It is clear that DNA methylation
can provide important knowledge that can be used for tumor early diagnosis, therapeutic decisions and risk assessments, and monitoring and surveillance [7]. Recent advances in sequencing provided great opportunities for quick analysis of epigenetic modifications in cancer cells. In return, the list of genes regulated by DNA methylation is quickly growing [10].
Chapter 2 Literature Review

The aim of the study is to extract associations between methylated genes and diseases automatically from free text. The purpose of the literature review is to investigate available computational resources related to gene methylation and diseases and any two entity in general. Firstly, section 2.1 Associations extraction tools from biomedical literature presents a survey of current association extraction tools. Next, section 2.2 Tools that accept user-provided text presents examples of some tools that process text submitted by users. Then section 2.3 Methylation databases provides information about several databases that include associations between methylated genes and diseases. Finally, section 2.4 Emerging issues and the need for further research summarizes how the previous work can be used to achieve the goals of this study.

2.1 Associations extraction tools from biomedical literature

Several tools have been developed in previous research with the aim to extract from text associations between biomedical entities such as drugs, genes, diseases, treatment side effects, transcription factors, etc. Basic information about some of these tools is summarized in Table 2-1. These tools are not available on the Web, but the table presents their reported performance and basic information about used datasets. The average accuracy and F-score of these tools are 79% and 51%-82%, respectively. Considering that these tools are concerned with topics different from link of gene methylation and diseases, these tools will not be used for performance comparison. However, it is important to review the state-of-the-art to know the level of performance of current (to an extent) similar systems.
Table 2-1: Summary of association extraction tools (ACC = accuracy, RE = recall and PR = precision)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Year</th>
<th>Dataset</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drug-drug interactions [22]</td>
<td>2012</td>
<td>Training: 5000 Pos. pairs, 5000 Neg. pairs</td>
<td>ACC = 79.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RE = 79.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SP = 78.9%</td>
</tr>
<tr>
<td>Drug-drug interactions [23]</td>
<td>2010</td>
<td>-</td>
<td>PR = 77.7% - 81.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>25.70%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PR = 67.30%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>48.69%</td>
</tr>
<tr>
<td>Drug-drug interactions [25]</td>
<td>2010</td>
<td>3775 sentences</td>
<td>RE = 82.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SP = 55.1%</td>
</tr>
<tr>
<td>Drug-drug interactions [26]</td>
<td>2011</td>
<td>3775 sentences</td>
<td>RE = 72.82%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PR = 51.03%</td>
</tr>
<tr>
<td>Gene–disease relations [27]</td>
<td>2006</td>
<td>1000 pairs</td>
<td>RE = 87.1%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PR = 78.5%</td>
</tr>
<tr>
<td>Gene–disease relations [28]</td>
<td>2006</td>
<td>2,042 pairs</td>
<td>PR = 70.75%</td>
</tr>
<tr>
<td>Gene–disease relations [29]</td>
<td>2003</td>
<td>1,000 sentences</td>
<td>PR = 76%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PR = 75% 27%</td>
</tr>
<tr>
<td>Gene–drug gene–disease relations [31]</td>
<td>2010</td>
<td>220 pairs</td>
<td>PR = 70-87.7%</td>
</tr>
<tr>
<td>Gene–drug relations [32]</td>
<td>2010</td>
<td>1731 articles</td>
<td>PR = 77.4%</td>
</tr>
<tr>
<td>Disease-drug relations [33]</td>
<td>2004</td>
<td>Training: 2793 sentences, Testing: 702 sentences</td>
<td>ACC = 79.6%</td>
</tr>
<tr>
<td>Disease-drug relations [34]</td>
<td>2008</td>
<td>Training: 2793 sentences, Testing: 702 sentences</td>
<td>ACC = 79.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PR = 73%</td>
</tr>
<tr>
<td>Clinical entities associations [37]</td>
<td>2008</td>
<td>77 documents</td>
<td>F-score = 72%</td>
</tr>
<tr>
<td>Gene–disease–drug relations [38]</td>
<td>2010</td>
<td>-</td>
<td>PR = 79–83%</td>
</tr>
<tr>
<td>Amino acid residues and their function in a protein relationships [39]</td>
<td>2003</td>
<td>1513 abstracts</td>
<td>RE = 82%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PR = 84%</td>
</tr>
<tr>
<td>Protein-protein interaction [40]</td>
<td>2003</td>
<td>4360 abstracts</td>
<td>PR = 20%</td>
</tr>
<tr>
<td></td>
<td><strong>Gene pathway relations [41]</strong></td>
<td>2004</td>
<td>Training = 40 Testing = 100 abstracts</td>
</tr>
<tr>
<td>---</td>
<td>--------------------------------</td>
<td>------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>21</td>
<td>Protein-protein interaction [42]</td>
<td>2009</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>Protein-protein interaction [43]</td>
<td>2011</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>Transcription factors and infectious diseases [43]</td>
<td>2011</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>Clinical entities relations [44]</td>
<td>2011</td>
<td>-</td>
</tr>
</tbody>
</table>

### 2.2 Tools that accept user-provided text

As examples we comment on two tools that can accept text submitted by users and extract associations from them. For example, Dragon TF Association Miner (DTFAM) [45] is a text mining tool that can detect if the PubMed abstracts contain explicit functional associations between transcription factors and Gene Ontology terms and/or disease terms. Abstracts are submitted by users. It has 82% and 80% sensitivity and specificity, respectively. It provides an interface to display graphical reports to link terms with relevant documents. Also, it allows users to summarize information to simplify analysis.

Dragon Plant Biology Explorer (DPBE) [46] is a text mining tool that can find associations between Arabidopsis genes and their functions from text submitted by users (see Figure 2-1). This tool is useful for users who want to summarize large information about genes or pathways. DPBE can identify potential novel associations between entities from several domain specific dictionaries. It can be used to create an information base on found associations.
2.3 Methylation databases

There have been efforts to collect associations between methylated genes and diseases and save them in databases. We present here four databases that have been manually curated, and the fifth one that is created by the automated process.

DiseaseMeth [47] is a large-scale database that includes experimental information about gene methylation in more than 50 diseases from literature and collected from various
websites. This database can be mainly used to search for associations of methylated genes and diseases. Users can also download, view and analyze the results of their search queries (see Figure 2-2).

PubMeth [48] is a database, but it is different from the previous one in a way that it only includes genes that are methylated in cancer. Users can use the database by submitting queries using a gene name (or names) only to obtain a list of associated cancers, or using cancer name (or names) only to get a list of genes which reported to be methylated. The tool also provides an option to summarize the results, which is a very good feature in case more than one gene or disease is searched at the same time.

MethyCancer [49] is similar to the previous database in a way that it only includes information collected from public resources about genes methylated in cancers. However, this database includes other information such as gene mutation and cancer type. It allows for presenting and analyzing interconnections between data. It has a user-friendly interface that gives access to all data, which makes it a very powerful tool (see Figure 2-3).

MethDB [50, 51] is a very special database as it provides information about gene methylation in 160 different tissues and 46 different species including humans. It facilitates query generation by providing options in a friendly interface, and it allows presenting the data in different ways (see Figure 2-4). Additionally, it includes an option that permits users to enter new data to the database.
Figure 2-2: DiseaseMeth Search Engine (from http://bioinfo.hrbumu.edu.cn/diseasemeth)
Figure 2-3: MethyCancer Search Engine (from http://methycancer.genomics.org.cn)

Figure 2-4: MethDB Search Engine (from http://www.methdb.net)
MeInfoText 2.0 [52] is also a database that only presents information about methylated genes and cancer associations. This database performs similar functions as the previous databases, but the associations are extracted automatically. The program implemented to extract associations has reported precision and recall of 91.8% and 90.0%, respectively, but the program is not available and thus the accuracy of the extraction method cannot be independently evaluated. Moreover, this database cannot be used to extract associations from text submitted by users.

2.4 Emerging issues and the need for further research

The importance of research in automated information extraction related to methylated genes and diseases becomes more apparent when other researchers try to collect such information. Manually-curated databases provide a good resource for partial information about methylated genes and their links to diseases. The fact that there are more than one database shows that neither one provides complete information. Thus, there is a need to have a tool that can extract such information in an automated fashion from large volume of documents and from the most current literature.

To the best of our knowledge, there is no publically available tool that extracts associations between methylated genes and diseases automatically from text submitted by users. In PubMed only there are more than 27,000 relevant documents. Also, there is no publically available annotated genes, methylation and diseases dataset which can be used for training and testing a machine learning model to allow for extracting information on association of methylated genes and diseases. This study makes contribution to providing such a dataset, a recognition algorithm that extracts the association between methylated
genes and diseases, and also a tool that allows users to submit text and extract such associations from the text.
Chapter 3 Methodology

This chapter aims to illustrate the steps that were followed in this study. Firstly, section 3.1 Problem Formulation discusses the first step that defines the problem studied. It also defines some terms that are used continuously throughout the paper. Secondly, section 3.2 Dataset explains the preparation of the dataset used in the study. Then, sections 3.6, 3.4 and 3.5 explain the methods used to process the data, to recognize the associations between the methylated genes and diseases, and other implementation issues. Finally, section 3.6 System Structure summarizes the developed system. The details about how the recognition model is developed based on comparison analysis of different machine learning approaches are presented in Chapter 4.

3.1 Problem Formulation

In order to extract associations between methylated genes and diseases from text, the following tasks have to be solved:

- Split text into sentences.
- Identify genes, diseases, and methylation words that are mentioned in a sentence.
- Using machine learning model predict if an identified gene in a sentence is methylated and associated with an identified disease in the sentence.

In this study, we have three classes of ‘concepts’, which refer to genes, diseases and methylation words. In our study, we consider only the cases where the concepts from these three classes exist in the same sentence in order to extract the associations. We use the term ‘pattern order’ to refer to the order of how these three concepts appear in sentences. Since we have three concepts then they can appear in sentences in six different
orders as illustrated in Table 3-1. We also use the term ‘pattern’ to indicate an instance of a pattern order. For example, ‘The CPG island in the FILIP1L promoter was heavily methylated in ovarian cancer cells’ (PMID 3157597) contains a gene ‘FILIP1L’, a disease ‘ovarian cancer’ and a methylation word ‘methylated’. So the pattern is (FILIP1L, methylated, ovarian cancer), but the pattern order is (<Gene>, <Methylation word>, <Disease>).

Table 3-1: Illustration of pattern order

<table>
<thead>
<tr>
<th>...</th>
<th>First Concept</th>
<th>...</th>
<th>Second Concept</th>
<th>...</th>
<th>Third Concept</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ...</td>
<td>&lt;Disease&gt;</td>
<td>...</td>
<td>&lt;Gene&gt;</td>
<td>...</td>
<td>&lt;Methylation word&gt;</td>
<td>...</td>
</tr>
<tr>
<td>2 ...</td>
<td>&lt;Disease&gt;</td>
<td>...</td>
<td>&lt;Methylation word&gt;</td>
<td>...</td>
<td>&lt;Gene&gt;</td>
<td>...</td>
</tr>
<tr>
<td>3 ...</td>
<td>&lt;Gene&gt;</td>
<td>...</td>
<td>&lt;Disease&gt;</td>
<td>...</td>
<td>&lt;Methylation word&gt;</td>
<td>...</td>
</tr>
<tr>
<td>4 ...</td>
<td>&lt;Gene&gt;</td>
<td>...</td>
<td>&lt;Methylation word&gt;</td>
<td>...</td>
<td>&lt;Disease&gt;</td>
<td>...</td>
</tr>
<tr>
<td>5 ...</td>
<td>&lt;Methylation word&gt;</td>
<td>...</td>
<td>&lt;Disease&gt;</td>
<td>...</td>
<td>&lt;Gene&gt;</td>
<td>...</td>
</tr>
<tr>
<td>6 ...</td>
<td>&lt;Methylation word&gt;</td>
<td>...</td>
<td>&lt;Gene&gt;</td>
<td>...</td>
<td>&lt;Disease&gt;</td>
<td>...</td>
</tr>
</tbody>
</table>

The model is supposed to find associations in the form:

\[
<\text{Gene}> \xrightarrow{\text{methylation word}} <\text{Disease}>
\]  

(1)

So in the previous example, the program should extract the following association:

\[
\text{FILIP1L} \xrightarrow{\text{methylated}} \text{Ovarian Cancer}
\]  

(2)

The relation between genes and diseases is a many-to-many relationship; one gene can be found to be methylated in several diseases, and one disease can be linked to several methylated genes. For example, BRCA1 is a gene and is found to be methylated in both
breast and ovarian cancers. Also, ovarian cancer is a disease and is linked to many methylated genes including genes BRCA1 and FILIP1L.

We formulate the problem we intend to solve as a binary classification problem. We use the term ‘positive sentences’ to refer to sentences that express associations between methylated genes and diseases and ‘negative sentences’ to refer to those that do not express such associations. So, the previous example of the sentence involving FILIP1L is the positive sentence because it conveys an association between FILIP1L and ovarian cancer, and therefore, the pattern (FILIP1L, methylated, ovarian cancer) is called a ‘positive pattern.’ However, the sentence ‘This study aimed to assess the impact of BRCA1 promoter methylation with respect to breast cancer subtypes’ (PMID 21593597) is the negative sentence, because it does not convey any association at all. In this sentence, ‘BRCA1’ is a gene, ‘breast cancer’ is a disease, and ‘methylation’ is a methylation word. Hence the pattern (BRCA1, methylation, breast cancer) represents a ‘negative pattern’ derived from the sentence.

Some sentences may contain more than one pattern, and it is possible to have positive and negative patterns in the same sentence. For example, in the sentence ‘We found that methylation of the T-cadherin promoter was present in 40% of HCC, but absent in liver cancer’ (PMID 18425332), ‘T-cadherin’ is a gene, ‘HCC’ and ‘liver cancer’ are diseases and ‘methylation’ is a methylation word. The pattern (methylation, T-cadherin, HCC) is a positive pattern because T-cadherin is associated with HCC whereas the pattern (methylation, T-cadherin, liver cancer) is a negative pattern because, based on this
sentence, *T-cadherin* is not associated with *liver cancer*. The model should identity both positive and negative patterns.

### 3.2 Datasets

The first set consists of 1540 abstracts from PubMed database [53]. From these abstracts we compiled set A that contains 2049 sentences with a total of 4662 different patterns since some sentences contained more than one pattern. These patterns were then classified into positive or negative patterns manually. 51% of patterns are positive, whereas the remaining 49% are negative. 30% of the sentences (570 sentences) form set P and was used for generating position weight matrices (PWMs) needed to develop the our classification model. The remaining 70% of sentences (1479 sentences with 3609 patterns: 1858 positive patterns and 1751 negative patterns) form the set C and was used for 10-fold cross-validation. Additionally, we compiled another set, set T, that consists of 200 patterns (100 positive and 100 negative patterns) from 75 sentences not part of the 2049 sentences described above.

### 3.3 Data Preprocessing

These tasks include sentence boundary determination, named entity recognition, tokenization, stemming, and keywords selection as explained in the following six subsections.

#### 3.3.1 Sentence Boundary Determination

Determining the sentence boundary (i.e., the end of a sentence) is a very important step because the program is supposed to extract associations from one sentence at a time. However, the data we used consists of scientific literature, and determining the sentence
boundary though looks trivial can be a tricky step. For example, a period can be used in
decimal numbers (such as 1.02), at the end of abbreviations (such as e.g.) or as an
indicator of the end of the sentence. We used a set of rules to identify End Of Sentences
(EOS) [54]:

- All ? and ! are EOS
- If “ or ‘ appears after a period, it is EOS
- If ) } or ] appear before a period, it is EOS
- If a line break or EOF (end of file) appears after a period, it is EOS
- If the token to which the period is attached has another period, it is not EOS
- If the token to which the period is attached begins with a lowercase letter and the
  next token begins with an uppercase letter, it is EOS
- Otherwise, the period is not EOS

3.3.2 Tokenization

Tokenization is a process that aims “to break the stream of characters into words” [54].
These words are called tokens, and each token belongs to a different type. Tokenization is
a very important step as it simplifies further extraction of information from a sequence of
characters. Tokenization of general text can be as easy as using whitespaces as words
delimiters. However, tokenization of scientific literature is not an easy task because of the
used domain specific language. Therefore, the following rules are applied to determine
the Boundary Of Words (BOW):

- Newline, tab, space, !, and ? are always BOW
- Period followed by whitespace is BOW
• If the word to which ““, ‘ ( ), [ ] or < > is attached at the end or the beginning does not include the matching punctuation in the middle of the word, it is BOW

• Otherwise, it is not BOW

The third rule is especially important because some gene names include punctuation as a part of the name. For example P14(ARF) is a gene, and, in this case, the parenthesis are considered as part of the gene name and cannot be used as a boundary of a word.

3.3.3 Named Entity Recognition

The tokenization step is very important to identify boundary of individual words. However, biomedical terms including genes and disease names usually consist of more than one word such as ‘GATA binding protein 2’. Therefore, identifying the boundary of multiword terms is handled by a separate step called named entity recognition. In this study, we used a dictionary-based approach to identify names of genes, diseases, and methylation words.

We used three dictionaries: genes, diseases and methylation words. These dictionaries include all various ways a word can be expressed. For example, the diseases dictionary includes ‘Diabetes Type 1’, ‘Diabetes Type1’, ‘Diabetes Type-1’, ‘Type1 Diabetes’, ‘Type 1 Diabetes’ and ‘Type-1 Diabetes.’ This is important in order to increase the recall rate of named entity recognition step.

3.3.4 Stop-words Elimination

Stop-words list is a compiled list of common words that can be discarded because they are likely not contributing to the discriminating features of the classes considered and thus not contributing to the predictive capability of a prediction model [54]. For example,
articles such as *a, an or the*, and pronouns such as *it, we* and *they* are considered as stop-words. This step must be performed after named entity recognition step because some genes or diseases names include stop-words. For example, ‘*Tumor of skin*’ is a disease and ‘*of*’ is a stop-word and must be recognized as part of the disease name. Eliminating stop-words early can hamper the named entity recognition step.

### 3.3.5 Stemming or Lemmatization

Stemming is the process in which each token is converted to a standard format [54]. In this process all suffixes and prefixes are removed from words, and words are replaced with their roots. This step is very important because it maps several words to one term and reduces the number of terms in the dataset. Also this can increase the frequency of occurrence of each term, which is important especially if we have a small dataset. Porter Stemmer [55] is used for this step. For the study, genes, diseases and methylation names are not stemmed, because stemming may change the meaning of these names. For example, *depression, depressing, and depressed*, after stemming they all will be converted to *depress*. However, the program should only identify *depression* as a disease but not *depressing or depressed*. Stemming genes, disease and methylation words can increase the ambiguity of the text, so these names are not stemmed. However, the rest of the words in the dataset are stemmed.

### 3.3.6 Keywords Selection

The tokenization step can produce a large number of tokens or words. Keywords are “a sequence of one or more words [that] provide a compact representation of a document’s content … and represent in a condensed form the essential content of a document” [56].
For our association recognition problem, a set of words that appear in a dataset contains words that may be of great value for that recognition and others that are not. Reducing the size of that set of words by eliminating those words that are of small value for our problem can be helpful as we use it in the process of model building and this can speed up the learning process and improve the quality of classification models. The reduced set of words identified from text we call keywords dictionary.

The previous two steps stop-words elimination and stemming are two mechanisms for reducing the size of the keyword dictionary. Also, we used information gain (IG) which estimates the information gained for predicting a class knowing a certain term is present or absent in a sentence [56]. Specifically, IG for a term \( t \) in class \( c \) when there are \( m \) classes is computed as follows [57]:

\[
IG(t) = -\sum_{i=1}^{m} P(c_i) \log P(c_i) + P(t) \sum_{i=1}^{m} P(c_i | t) \log P(c_i | t) + P(\overline{t}) \sum_{i=1}^{m} P(c_i | \overline{t}) \log P(c_i | \overline{t})
\]

where \( P(t), P(\overline{t}), P(c_i | t) \) and \( P(c_i | \overline{t}) \) is the probability that \( t \) appears, \( t \) does not appear, the class is \( c_i \) given \( t \) appeared, and the class is \( c_i \) given \( t \) does not appear, respectively. A term is considered a keyword if its IG is greater than a certain threshold.

### 3.4 Structured Data Representation

One of the important steps in text mining is to transform the text from free unstructured format to a structured form. Mainly, we used two transformation mechanisms where each data structure provides different types of statistical information. The first approach is based on DTFM (document-term frequency matrix), and the second approach on PWMs (Position Weight Matrices).
3.4.1 Document-Term Frequency Matrix (DTFM)

The first approach is based on using DTFM. In DTFM each row corresponds to a sentence, and each column corresponds to a keyword [54]. Keywords are determined by using the keyword dictionary. The cells are represented using there mechanisms (see Figure 3-1):

- Binary values: each cell has a value 0 or 1 to indicate if a word appears in a sentence or not, respectively.
- Frequency values: a cell represents the frequency of a word in a sentence.
- Term-frequency inverse-document-frequency (TF-IDF) as follows [56]:

\[
TF-IDF(w,d) = f(w,d) \times (\log \frac{N}{n} + 1)
\]

where \( f(w,d) \) is the frequency of word \( w \) in document \( d \), and \( N \) is the total number of sentences, and \( n \) is the number of sentences where the word \( w \) appears. We used z-score normalization for attribute \( A \) where \( v \) is the original value of the attribute, \( \bar{A} \) is the mean, and \( \sigma_A \) is standard deviation as follows [58]:

\[
v' = \frac{v - \bar{A}}{\sigma_A}
\]
3.4.2 Position Weight Matrices (PWMs)

PWMs can represent the statistical distribution of frequency of occurrence of certain elements relative to specific positions. PWMs have been used widely in bioinformatics for several purposes such as binding sites identification [59], motifs discovery [60], etc.

For example, below we show a set of DNA sequences [61].

<table>
<thead>
<tr>
<th>Sequence1</th>
<th>G</th>
<th>A</th>
<th>G</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence2</td>
<td>T</td>
<td>C</td>
<td>C</td>
<td>G</td>
<td>T</td>
<td>A</td>
<td>A</td>
<td>G</td>
<td>T</td>
</tr>
<tr>
<td>Sequence3</td>
<td>C</td>
<td>A</td>
<td>G</td>
<td>G</td>
<td>T</td>
<td>T</td>
<td>G</td>
<td>G</td>
<td>A</td>
</tr>
<tr>
<td>Sequence4</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>G</td>
<td>T</td>
<td>C</td>
<td>A</td>
<td>G</td>
<td>T</td>
</tr>
<tr>
<td>Sequence5</td>
<td>T</td>
<td>A</td>
<td>G</td>
<td>G</td>
<td>T</td>
<td>C</td>
<td>A</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

PWM generated from these sequences has four rows, because there are four different entities (nucleotides denoted as A, C, G and T), and nine columns because of the length
of the sequences. For example, in the first column in the table above, we see that A, C and G appear only once whereas T appears twice, and these values are stored in the first column of the position frequency matrix below.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3-3: Position Frequency Matrix

After computing the frequencies, the matrix is normalized to get the PWM as presented in Table 3-4. In the previous table, the value of one cell in a certain column is divided by the sum of frequencies of that column. The sum of each column is five, so each cell content is divided by five.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.2</td>
<td>0.6</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>0.8</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>C</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>G</td>
<td>0.2</td>
<td>0</td>
<td>0.6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0.4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.2</td>
<td>0</td>
<td>0.2</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 3-4: Position Weight Matrix

3.4.3 PWMs Generation from Biomedical Literature

We used the concept of PWM to capture characteristics of text for use in our machine learning models. The sentences are aligned along the three concepts (genes, diseases and methylation words). This way, we identified four different positions for each sentence. Since there are six different orderings of these concepts (as illustrated in Table 3-1), we have six PWMs each corresponding to a certain pattern order (see Figure 3-2).
There are four columns in the PWMs, and each row corresponds to a certain word. Each column represents a certain position in the sentence. For example, in the first PWM in Figure 3-2, the first position represents the frequency of the words that appear before the disease concept. Similarly, the second, third and fourth positions represent the frequency of words that appear between disease and gene concepts, gene and methylation words concepts, and after methylation words concept respectively.

![Position Weight Matrices](image)

**Figure 3-2: PWMs**

PWMs can capture characteristics of one class of data only. Therefore, we generated 12 PWMs: six PWMs for the positive sentences (called ‘positive PWMs’), and other six PWMs for the negative sentences (called ‘negative PWMs’). We have two dictionaries: a ‘positive dictionary’ that contains words for the positive class, and a ‘negative dictionary’ that contains words for the negative class. The frequency of a word in one class is computed by counting the number of sentences in which a word appears in that class.
class divided by the total number of sentences in that class. If a word is more frequent in the positive class, then it is a word for the positive dictionary; otherwise it is a word for the negative dictionary. The number of rows in the positive and negative PWMs is the number of distinct words in positive and negative dictionaries, respectively. To generate the PWMs, we use collection of positive examples or negative examples, as follows:

1. Determine the pattern that appears in the sentence.
2. Determine the pattern order.
3. Identify words in each position.
4. Update the PWM that corresponds to the pattern order.

For example, from the sentence ‘The CPG island in the FILIP1L promoter was heavily methylated in ovarian cancer cells’ (PMID 3157597) we find the pattern as (FILIP1L, methylated, ovarian cancer) and the pattern order as (<gene>, <methylation word>, <disease>). Therefore the PWM that corresponds to this pattern order will be updated (see Figure 3-3). The words (CPG, island) appear in the first position, so the cells that correspond to the rows of these words and the first column of the PWM will be incremented by one. Similarly the same step is applied for the remaining three positions.

![Figure 3-3: PWMs Generation Example](image-url)
3.4.4 Computing the Scores from PWMs

After generating the PWMs, the matrices are normalized by dividing the frequency of each word by the total frequency along each column. We use these PWMs to generate the matching scores for each of the sentences. The scores are determined following the steps below:

1. Determine the pattern that appears in the sentence.
2. Determine the pattern order.
3. Get the PWM that corresponds to this pattern order.
4. Identify words in each position.
5. If there are many words in one position, get the maximum weight in each position.
6. Sum the weights of the four positions to get the final score of the sentence.

Assume that there are several pattern orders in one sentence. It is likely that more than one PWM will give a score to the sentence. For example ‘promoter of MIR203 was found methylated in approximately 25% MM cell lines but not methylated in normal controls,’ has one pattern in two orders (MIR203, methylated, MM) and (MIR203, MM, methylated), so we will get four scores from two positive PWMs and two negative PWMs that correspond to the two pattern orders. In general, we can get up to twelve scores for each sentence.

Figure 3-4 demonstrates how to compute the score for the sentence in previous example with respect to the first pattern. Each position in the sentence is highlighted by a different color. There is only one keyword in each position except for the last position that contains five keywords. In this case, when there is more than one word in a certain
position, we take the maximum weight to compute the score. The final score can be computed as follows:

$$0.2336 + 0.0619 + 0.1724 + 0.1315 = 0.5994$$  \hspace{1cm} (6)

We should mention that there are other ways to compute score in the cases of multiple words found on the same position. For example, one can sum up all the weights of these words. However, this resulted in low performance, so we used the method explained above.

The score for the second pattern can be computed in the same way by using another PWM that corresponds to the other pattern order.

![Position Weight Matrix](image)

**Figure 3-4: Example for Computing the Score**

### 3.4.5 Feature Generation

We represent each sentence with twelve features and a label. The twelve features correspond to the scores taken from PWMs. For each sentence, we compute six scores

Promoter of *MIR203* was found *methylated* in approximately 25% of *MM* cell lines but not methylated in normal controls.
from the six positive PWMs and other six scores from the negative PWMs. Figure 3-5 illustrates how the scores are used as features for the dataset.

<table>
<thead>
<tr>
<th>Six Scores from Positive PWMs</th>
<th>Six Scores from Negative PWMs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0.2920</td>
</tr>
<tr>
<td>S2</td>
<td>0.2482</td>
</tr>
<tr>
<td>S3</td>
<td>0.2482</td>
</tr>
<tr>
<td>S4</td>
<td>0.2409</td>
</tr>
<tr>
<td>S5</td>
<td>0.2409</td>
</tr>
<tr>
<td>S6</td>
<td>0.2366</td>
</tr>
</tbody>
</table>

**Figure 3-5: Illustration of Features Generated by PWMs**

This approach can be used to represent a sentence in multiple ways depending on the patterns. For example, in this sentence ‘We found that methylation of the T-cadherin promoter was present in 40% of HCC, but absent in liver cancer’ (PMID 18425332), the pattern P1 (methylation, T-cadherin, HCC) is a positive pattern whereas the pattern P2 (methylation, T-cadherin, liver cancer) is a negative pattern as explained in Section 3.1.

The features of each pattern are shown in Table 3-5.

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0511</td>
</tr>
<tr>
<td>0</td>
<td>0.0690</td>
</tr>
<tr>
<td>0</td>
<td>0.0690</td>
</tr>
<tr>
<td>0</td>
<td>0.0219</td>
</tr>
<tr>
<td>0</td>
<td>0.0219</td>
</tr>
<tr>
<td>0</td>
<td>0.0219</td>
</tr>
<tr>
<td>0</td>
<td>0.0102</td>
</tr>
<tr>
<td>0</td>
<td>0.0102</td>
</tr>
<tr>
<td>0</td>
<td>0.0167</td>
</tr>
<tr>
<td>0</td>
<td>0.0167</td>
</tr>
<tr>
<td>0</td>
<td>0.0123</td>
</tr>
<tr>
<td>0</td>
<td>0.0123</td>
</tr>
<tr>
<td>0</td>
<td>0.0331</td>
</tr>
<tr>
<td>0</td>
<td>0.0331</td>
</tr>
<tr>
<td>0</td>
<td>0.0185</td>
</tr>
<tr>
<td>0</td>
<td>0.0185</td>
</tr>
<tr>
<td>0</td>
<td>0.0123</td>
</tr>
<tr>
<td>0</td>
<td>0.0123</td>
</tr>
<tr>
<td>0</td>
<td>0.0123</td>
</tr>
<tr>
<td>0</td>
<td>0.0123</td>
</tr>
</tbody>
</table>

This way we will be able to describe different patterns by set of features that in principle have different values. Consequently, such representation is useful for classifying each
pattern as positive or negative even though they appear in the same sentence. Interestingly, separation of positive and negative patterns from the same sequence cannot be done by the standard DTFM approach.

### 3.5 Classification

In the search for the most efficient machine learning model we compared performances of several different classifiers on our data. The classifiers included rule-based ones, decision trees, support vector machines (SVM), K nearest neighbor (KNN), and Naïve Bayes. They are briefly explained in the following subsections.

#### 3.5.1 First Order Inductive Learner (FOIL)

FOIL [62] is an algorithm for generating classification rules. FOIL generates rules by dividing a training set into two sets: $P$ contains positive records and $N$ contains negative records, and generates rules for each set separately. FOIL depends on computing the gain measure every time a new attribute is added to the antecedent of the rule as follows [63]:

$$Gain = p \left( \log_2 \frac{p}{p+n} - \log_2 \frac{p'}{p'+n'} \right)$$

(7)

where $p$ and $n$ are the number of positive and negative records covered by the new rule respectively, and $p'$ and $n'$ are the number of positive and negative records covered by the parent rule respectively. FOIL keeps adding an attribute to the rule’s antecedent until the gain value becomes below a user-specified minimum value, the number of attributes in the antecedent is greater than a user-specified maximum value, or the rule does not cover any record in the opposite class [64]. The records that are matched by the rule are then deleted from the dataset. FOIL generates more rules by using the remaining training
set. FOIL does not generate a large number of rules, so no rules pruning is required. The final rules set is ordered by using the Laplace measure as follows [63]:

\[
Laplace = \frac{p + 1}{p + n + 2}
\]

(8)

During testing, the best \( k \) rules whose antecedent is a subset of the record are selected. Then the class that has greater average will be predicted for the record.

3.5.2 Predictive Rule Mining (PRM)

PRM [65] an algorithm for generating association rules, and it is a modified version of FOIL. PRM gives weights for records in the dataset. Unlike FOIL, PRM does not remove records from the dataset which are covered by a rule. Instead, weights for these records are reduced by a decay factor. The records are removed only when the weight is below a threshold. Initially each record has a weight equals to one, and the new weight is computed by multiplying the old weight with the decay factor [66].

The threshold is computed by multiplying the start weigh of the records by a user specified total weight factor. The gain value is calculated by summing the weight values of the sentences covered by the rule instead of counting the number of these sentences. Unlike FOIL, PRM does not require setting a maximum number of attributes in each rule. It is possible that some rules will have the same antecedent, so the rules that have the lower Laplace value will be deleted from the rules set [66].

3.5.3 Classification based on Predictive Association Rules (CPAR)

CPAR [65] an algorithm for generating classification rules, and it is a modified version of PRM. CPAR chooses several attributes if they have the same best gain value instead of
choosing one attribute only. All attributes that have gain better than best gain multiplied by similarity threshold are selected. The similarity threshold is specified by users [67].

3.5.4 Classification based on Associations (CBA)

CBA [68] is a rule generation algorithm, and it operates on two stages: frequent item set selection by using support threshold, and rules pruning by using confidence threshold. Frequent item set is a subset of attributes whose support is greater than the threshold. The support and confidence are computed as follows:

\[
Support = \frac{\text{# of sentences covered by the frequent item set}}{\text{total number of sentences}}
\]

(9)

\[
Confidence = \frac{\text{# of sentences covered by the rule}}{\text{# of sentences covered by the antecedent of the rule}}
\]

(10)

CBA begins by generating a set of possible rules ordered according to higher confidence, higher support then lower number of attributes. Then CBA identifies all correct rules and wrong rules, and records the number of records covered by strong correct rules. A correct rule is a strong rule if it has higher precedence than the corresponding wrong rule. Next, CBA processes records that were wrongly predicted. A wrong rule can be removed, if it is not a strong correct rule of another record. For every strong correct rule, CBA determines the default class and total error. Total error is computed by summing the number of wrongly predicted records by the rules and the number of wrongly predicted records by the default rule. The majority class in the training set is considered the default class for the default rule [69].
3.5.5 Classification based on Multiple Association Rules (CMAR)

CMAR [70] is a rule generation algorithm, and it operates on two stages as CBA, but CMAR uses FP growth algorithm to generate the rules. A rule is considered as a classification rule if its Chi-Square value is above a threshold and there is no other general rule with a higher priority. A rule is more general than another rule if its antecedent is a superset of the antecedent of the other rule. Higher confidence and support and lower number of attributes give a rule more priority. To test the rules, if the rules that cover the testing record have the same class, the class will be used. Otherwise, group the rules according to the class, and select the group that has the highest Weighted Chi Square (WCS). WCS for each group is computed as follows [71]:

\[
WCS = \frac{\text{the sum of (Chi-Square} \times \text{Chi-Square})}{MCS}
\]  \hspace{1cm} (11)

where MCS is Minimum Chi-Square and computed as follows [71]:

\[
MCS = \left[ \min(\text{sup}(A), \text{sup}(c)) - \frac{\text{sup}(A)\text{sup}(c)}{N} \right]^2 \times N \times e
\]  \hspace{1cm} (12)

where sup(A), sup(c) and N are support of antecedent, support of consequent and the number of records in test set respectively, and e is computed as follows [71]:

\[
e = \frac{1}{\text{sup}(A)\text{sup}(c)} + \frac{1}{\text{sup}(A)N - \text{sup}(c)} + \frac{1}{N - \text{sup}(A)\text{sup}(c)} + \frac{1}{(N - \text{sup}(A))(N - \text{sup}(c))}
\]  \hspace{1cm} (13)

3.5.6 Total from Partial Classification (TFPC)

TFPC [72] is a rule generation algorithm, and it depends on computing support and confidence to determine a classifier similar to CMAR. However, TFPC is based on a T-tree (Total-support tree) that keeps track of frequent item sets. Each level of the tree
includes several nodes, and each node has two components: the support, and a pointer to the next node in the tree [73]. Unlike CMAR, classification rules are generated in the same stage of generating frequent item set.

### 3.5.7 C4.5

C4.5 grows trees by using divide and conquer approach [74]. Each node in the tree is assigned a set of records, and every record is assigned a weight to consider unknown attributes [75]. Each leaf contains the class. The weighted frequency \( \text{freq}(C_i, T) \) is computed where \( T \) is the set of records at a node, and \( C_i \) is the class of the records. If all records in the node are of the same class \( C_i \), or the number of records in the node is less than a certain threshold, then the node is a leaf. The classification error at leaves is the weighted sum of records in the node that are not of class \( C_i \) [76]. If \( T \) contains two or more classes, information gain is computed, and the attribute that has the best information gain is used for splitting at the node. Information gain for an attribute \( a \) for a set of records \( T \) is computed as follows:

\[
gain = \text{info}(T) - \sum_{i=1}^{s} \left| \frac{T_i}{|T|} \right| \times \text{info}(T_i)
\]

where \( s \) is the number of distinct values, and \( T_i \) is the subset of record that has the same value of \( a \), and the entropy function is:

\[
\text{info}(T) = -\sum_{j=1}^{\text{NClass}} \frac{\text{freq}(C_j, T)}{|T|} \times \log_2 \left( \frac{\text{freq}(C_j, T)}{|T|} \right)
\]

C4.5 computes the information gain ratio as follows:
\[
Split(T) = - \sum_{i=1}^{k} \frac{|T|}{|T|} \times \log_2 \left( \frac{|T_i|}{|T|} \right)
\]  

Finally, the classification error of the node is computed by summing classification errors of all child nodes. If the error is greater than the error of classifying the records with the frequent class in the node, then the node will be considered a leaf and all sub-trees are deleted.

3.5.8 Random tree

Random tree models have been studied in machine learning field in recent years [77]. Several ways can be used to generate different models of random tree with \( n \) vertices. A random tree is a tree that is selected randomly from the possible \( n^{n-2} \) trees on vertices \( 1, \ldots, n \) [78]. Each tree is equally likely to be sampled from the set of trees. Each node has \( k \) features selected randomly [77].

3.5.9 Random Forest

Random Forest consists of several random trees. To use random forest for classifications, each tree gives a classification vote for the input record, and the final class of the record is determined by taking majority voting. In the original paper of random forest [79], Breiman shows that random forest can handle a big number of features and provide good estimates of what features can be considered for classification. Also, random forest is generally efficient, and there is no imposed limitation to the number of generated trees. Additionally, random forest do not over-fit, and the generated trees can be saved to be used on other data. To construct the trees, consider that we have \( N \) training samples and \( M \) number of variables. Select \( n \) number of samples with replacement from the training
set where \( n \) is smaller than \( N \); these \( n \) samples will be used for training the tree, and the rest of the samples are used for testing. Select \( m \) variables where \( m \) is smaller than \( M \) to determine the decision at each node in the tree; these \( m \) variables are used to calculate the best split. All trees are fully grown, so there is no pruning [80].

3.5.10 Support Vector Machines (SVM)

SVM [81] searches for an optimal hyper-plane to separate between two classes of data. SVM tries to increase the margin between closest points of the classes. The optimal hyper-plane is the middle of the margin whereas support vectors are the points that reside on the boundaries. When the dataset cannot be separated linearly, the points are projected to a higher dimensional space by using kernels [82].

3.5.11 K-nearest neighbors (KNN)

KNN is a very simple and one of the earliest data classification algorithm [83]. It classifies unseen records by taking majority voting among its nearest \( k \) records from the training set. Its performance is mainly dependent on \( k \) and the distance measure. If there is no prior knowledge of the data, it can be hard to choose an optimal distance measure. However, usually it provides competitive results [84].

3.5.12 Naïve Bayes

Naïve Bayes is a probabilistic classifier [85]. Bayes’ theorem is as follows:

\[
P(c_i | d_j) = \frac{P(c_i)P(d_j | c_i)}{P(d_j)}
\]  

(17)

where \( P(c_i | d_j) \) is the probability that a document \( d \) has a class \( c \). It is assumed that documents features are statistically independent, so \( P(d_j | c_i) \) is computed as follows [86]:
\[ P(d_j \mid c_i) = \prod_{k=1}^{n} P(w_{kj} \mid c_i) \] (18)

\( T \) is the set of features in the dataset.

### 3.6 System Structure

We developed a system that can analyze the user provided free text data and identify the associations between methylated genes and diseases. The system structure is presented in Figure 3-6. The recognition algorithm is the one that performed the best in comparison. The algorithm is based on random forest model.

![Figure 3-6: System Architecture](image-url)
Chapter 4 Experiments and Results

The aim of the chapter is to provide details on performed experiments that are used to select the best performing model for automated recognition of positive and negative examples for our problem. The experiments are described in terms of: algorithms, parameters, features and results that were produced by the three approaches: DTFM, PWMs and a hybrid approach. Firstly, section 4.1 Performance Metrics explains the metrics which are used to evaluate the performance of the algorithms. Then section 4.2 PWMs Approach shows the results by using PWMs approach. Consequently, section 4.3 DTFM Approach shows the results by using DTFM approach. After that, section 4.4 Hybrid Approach shows the results by using a hybrid approach. Next, section 4.5 Testing Results shows the results on the testing set of the best algorithm. Finally, the last section 4.6 Summary of Results summarizes all the results that are mentioned in the chapter.

4.1 Performance Metrics

We compute precision (PR), recall (RE), specificity (SP), accuracy (ACC) and F-score as follows:

\[
PR = \frac{TP}{TP + FP} \tag{19}
\]

\[
RE = \frac{TP}{TP + FN} \tag{20}
\]

\[
SP = \frac{TN}{TN + FP} \tag{21}
\]

\[
ACC = \frac{TP + TN}{TP + FN + TN + FP} \tag{22}
\]

\[
F - score = 2 \times \frac{PR \times RE}{PR + RE} \tag{23}
\]
If a pattern is positive and was predicted as positive then it is TP (true positive), but if it was predicted as negative, then it is FN (false negative). On the other hand, if a pattern is negative and is predicted as negative then it is TN (true negative), but if it was predicted as positive then it is FP (false positive).

4.2 PWMs Approach

We evaluated the performance of several algorithms using the features generated by PWMs. PWMs were generated from set P. The feature set consists of twelve features: six scores from the positive PWMs and six scores are from the negative PWMs. We applied 10-fold cross-validation with all the algorithms on set C. Set C contains 1479 sentences with 3609 patterns (1858 positive patterns and 1751 negative patterns). We used implementation of random forest, C4.5 and random tree on WEKA [87]. We used LIBSVM [88] toolbox in MATLAB [89] for SVM implementation. We used implementation of KNN from MATLAB. We tried different parameters for each algorithm. Table 4-1 shows the parameters that we used for each algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameter</th>
<th>values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>Number of trees</td>
<td>5, 10, 15, 20, 25, 30</td>
</tr>
<tr>
<td></td>
<td>Number of randomly selected features</td>
<td>2, 4, 6, 8, 10, 12</td>
</tr>
<tr>
<td>SVM</td>
<td>Cost, Coef0, Degree</td>
<td>$2^n$, $n = {1, 2, 3, 4, 5}$</td>
</tr>
<tr>
<td></td>
<td>Gamma</td>
<td>$\frac{1}{2^n}$, $n = {1, 2, 3, 4, 5}$</td>
</tr>
<tr>
<td>KNN</td>
<td>Number of nearest Neighbors</td>
<td>$2^n-1$, $n = {1, 2, 3, 4, 5}$</td>
</tr>
<tr>
<td>C4.5</td>
<td>Confidence</td>
<td>0.1, 0.3, 0.5, 0.7, 0.9</td>
</tr>
<tr>
<td>Random Tree</td>
<td>Number of randomly selected features</td>
<td>2, 4, 6, 8, 10, 12</td>
</tr>
</tbody>
</table>
We recorded the best performance with the corresponding parameters. Table 4-2 shows 10-fold cross-validation results after applying the algorithms on sentences that contain only one pattern from set C. In this case, the sentence is positive if the pattern is positive; otherwise, the sentence is negative if the pattern is negative.

Table 4-2: 10-fold cross-validation results on set C using sentences with one pattern only

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ACC</th>
<th>PR</th>
<th>RE</th>
<th>SP</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random forest</td>
<td>69.2%</td>
<td>74.2%</td>
<td>79.1%</td>
<td>51.7%</td>
<td>15 decision trees and 4 random features</td>
</tr>
<tr>
<td>SVM</td>
<td>70.3%</td>
<td>75.3%</td>
<td>80.2%</td>
<td>51.6%</td>
<td>Polynomial kernel cost = 8, gamma=0.25, coeff=8, degree=4</td>
</tr>
<tr>
<td>KNN</td>
<td>68.7%</td>
<td>71.5%</td>
<td>85.2%</td>
<td>49.7%</td>
<td>City block distance and 5 nearest neighbors</td>
</tr>
<tr>
<td>C4.5</td>
<td>70.4%</td>
<td>71.4%</td>
<td>89.3%</td>
<td>37.4%</td>
<td>Confidence = 0.1</td>
</tr>
<tr>
<td>Random Tree</td>
<td>66.8%</td>
<td>73.4%</td>
<td>74.9%</td>
<td>52.6%</td>
<td>4 random features selected</td>
</tr>
</tbody>
</table>

Table 4-3 shows 10-fold cross-validation results of algorithms when applied on sentences with multiple patterns from set C. The algorithms classify patterns independently even if they appear in the same sentence.

Table 4-3: 10-fold cross-validation results on set C using sentences with multiple patterns only

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ACC</th>
<th>PR</th>
<th>RE</th>
<th>SP</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random forest</td>
<td>85.5%</td>
<td>85.8%</td>
<td>85.0%</td>
<td>86.0%</td>
<td>10 decision trees and 12 random features</td>
</tr>
<tr>
<td>SVM</td>
<td>69.6%</td>
<td>69.9%</td>
<td>69.1%</td>
<td>70.1%</td>
<td>Polynomial Kernel cost = 8, gamma=0.5, coeff=8, degree=4</td>
</tr>
<tr>
<td>KNN</td>
<td>70.1%</td>
<td>67.9%</td>
<td>75.0%</td>
<td>65.9%</td>
<td>Euclidean distance and 3 nearest neighbors</td>
</tr>
<tr>
<td>C4.5</td>
<td>80.5%</td>
<td>79.4%</td>
<td>82.2%</td>
<td>78.8%</td>
<td>Confidence = 0.7</td>
</tr>
<tr>
<td>Random Tree</td>
<td>85.2%</td>
<td>85.0%</td>
<td>85.3%</td>
<td>85%</td>
<td>12 random features selected</td>
</tr>
</tbody>
</table>
Table 4-4 shows 10-fold cross-validation results after applying the algorithms on the set C that contains sentences with one pattern only and sentences with multiple patterns.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ACC</th>
<th>PR</th>
<th>RE</th>
<th>SP</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random forest</td>
<td>81.5%</td>
<td>81.2%</td>
<td>84.7%</td>
<td>77.8%</td>
<td>10 decision trees and 4 random features</td>
</tr>
<tr>
<td>SVM</td>
<td>69.6%</td>
<td>69.3%</td>
<td>72.8%</td>
<td>65.7%</td>
<td>Polynomial kernel cost = 8, gamma=0.5, coeff=8, degree=4</td>
</tr>
<tr>
<td>KNN</td>
<td>70.3%</td>
<td>69.7%</td>
<td>77.1%</td>
<td>63.4%</td>
<td>Euclidean distance and 3 nearest neighbors</td>
</tr>
<tr>
<td>C4.5</td>
<td>77.0%</td>
<td>77.2%</td>
<td>80.3%</td>
<td>73.2%</td>
<td>Confidence = 0.7</td>
</tr>
<tr>
<td>Random Tree</td>
<td>81.0%</td>
<td>80.3%</td>
<td>85.0%</td>
<td>76.4%</td>
<td>6 random features selected</td>
</tr>
</tbody>
</table>

### 4.3 DTFM Approach

We evaluated the performance of several algorithms using DTFM. To classify sentences that contain one pattern, a sentence is positive if the pattern is positive; otherwise, the sentence is negative if the pattern is negative. To classify sentences with multiple patterns, a sentence is positive if there is at least one positive pattern; otherwise, the sentence is negative if all patterns are negative. DTFM approach can be used to find if a sentence contains a positive pattern or not, but we do not know which pattern is the positive one. The following subsections give more details about the experiments.

#### 4.3.1 Rule Generation Algorithms

We used LUCS-KDD implementation of rule generation algorithms [64]. For FOIL, CPAR and PRM, we tried different gain values and recorded accuracy. We used binary values with DTFM. We applied 10-fold cross-validation on set C. Table 4-5 shows the accuracy after applying the algorithms on set C and using different gain values.
Table 4-5: 10-fold cross-validation accuracy after using different gain values on set C

<table>
<thead>
<tr>
<th>Gain Value</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
<th>80%</th>
<th>90%</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOIL</td>
<td>69%</td>
<td>69%</td>
<td>70%</td>
<td>71%</td>
<td>70%</td>
</tr>
<tr>
<td>CPAR</td>
<td>63%</td>
<td>62%</td>
<td>61%</td>
<td>62%</td>
<td>63%</td>
</tr>
<tr>
<td>PRM</td>
<td>61%</td>
<td>61%</td>
<td>61%</td>
<td>61%</td>
<td>62%</td>
</tr>
</tbody>
</table>

Table 4-6 shows the 10-fold cross-validation accuracy after applying the algorithms on sentences that contain only one pattern and using different gain values.

Table 4-6: 10-fold cross-validation accuracy after using different gain values on sentences with one pattern from set C

<table>
<thead>
<tr>
<th>Gain Value</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
<th>80%</th>
<th>90%</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOIL</td>
<td>51%</td>
<td>53%</td>
<td>52%</td>
<td>52%</td>
<td>52%</td>
</tr>
<tr>
<td>CPAR</td>
<td>41%</td>
<td>42%</td>
<td>43%</td>
<td>41%</td>
<td>42%</td>
</tr>
<tr>
<td>PRM</td>
<td>40%</td>
<td>41%</td>
<td>50%</td>
<td>41%</td>
<td>41%</td>
</tr>
</tbody>
</table>

For CBA, CMAR and TFPC, we tried different values for support and confidence (cnfd) to see their effects on the accuracy. We used binary values with DTFM. We applied 10-fold cross-validation. Table 4-7 and Table 4-8 summarize these results.

Table 4-7: 10-fold cross-validation accuracy of algorithms on set C

<table>
<thead>
<tr>
<th>support</th>
<th>cnfd</th>
<th>ACC</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TFPC</td>
<td>CMAR</td>
<td>CBA</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>40</td>
<td>74%</td>
<td>65%</td>
<td>77%</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>50</td>
<td>72%</td>
<td>68%</td>
<td>77%</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>60</td>
<td>70%</td>
<td>66%</td>
<td>76%</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>70</td>
<td>68%</td>
<td>66%</td>
<td>76%</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>80</td>
<td>73%</td>
<td>64%</td>
<td>74%</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>90</td>
<td>74%</td>
<td>56%</td>
<td>79%</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>40</td>
<td>74%</td>
<td>63%</td>
<td>78%</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>50</td>
<td>74%</td>
<td>67%</td>
<td>78%</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>60</td>
<td>73%</td>
<td>65%</td>
<td>78%</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>70</td>
<td>72%</td>
<td>64%</td>
<td>76%</td>
<td></td>
</tr>
</tbody>
</table>

Table 4-8: 10-fold cross-validation accuracy of algorithms on one pattern sentences on set C

<table>
<thead>
<tr>
<th>support</th>
<th>cnfd</th>
<th>ACC</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TFPC</td>
<td>CMAR</td>
<td>CBA</td>
</tr>
<tr>
<td>0.25</td>
<td>40</td>
<td>62%</td>
<td>32%</td>
<td>55%</td>
</tr>
<tr>
<td>0.25</td>
<td>50</td>
<td>59%</td>
<td>30%</td>
<td>55%</td>
</tr>
<tr>
<td>0.25</td>
<td>60</td>
<td>55%</td>
<td>30%</td>
<td>55%</td>
</tr>
<tr>
<td>0.25</td>
<td>70</td>
<td>57%</td>
<td>29%</td>
<td>55%</td>
</tr>
<tr>
<td>0.25</td>
<td>80</td>
<td>58%</td>
<td>24%</td>
<td>54%</td>
</tr>
<tr>
<td>0.25</td>
<td>90</td>
<td>59%</td>
<td>17%</td>
<td>52%</td>
</tr>
<tr>
<td>0.5</td>
<td>40</td>
<td>65%</td>
<td>33%</td>
<td>57%</td>
</tr>
<tr>
<td>0.5</td>
<td>50</td>
<td>64%</td>
<td>30%</td>
<td>58%</td>
</tr>
<tr>
<td>0.5</td>
<td>60</td>
<td>63%</td>
<td>29%</td>
<td>59%</td>
</tr>
<tr>
<td>0.5</td>
<td>70</td>
<td>62%</td>
<td>28%</td>
<td>59%</td>
</tr>
</tbody>
</table>
4.3.2 Decision Trees Algorithms

We used WEKA tool [87] for training and testing random forest, C4.5 and random tree.

We performed 10-fold cross-validation on set C. Firstly, we tried to determine the best number of keywords to consider as features. We used TF-IDF and z-score with DTFM.

Table 4-9 shows the accuracy of these algorithms after applying them on set C by selecting a different number of keywords. The keywords were selected by using information gain as explained in Section 3.3.6.

<table>
<thead>
<tr>
<th># of keywords</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5</td>
<td>65%</td>
<td>70%</td>
<td>71%</td>
<td>74%</td>
<td>74%</td>
<td>76%</td>
<td>75%</td>
<td>75%</td>
<td>75%</td>
<td>75%</td>
</tr>
<tr>
<td>Random forest</td>
<td>64%</td>
<td>70%</td>
<td>72%</td>
<td>77%</td>
<td>78%</td>
<td>80%</td>
<td>78%</td>
<td>78%</td>
<td>78%</td>
<td>78%</td>
</tr>
<tr>
<td>Random Tree</td>
<td>64%</td>
<td>70%</td>
<td>69%</td>
<td>72%</td>
<td>73%</td>
<td>73%</td>
<td>73%</td>
<td>73%</td>
<td>72%</td>
<td>73%</td>
</tr>
</tbody>
</table>

Table 4-9: 10-fold cross-validation accuracy of algorithms after applying them on set C
It is clear that all the three algorithms give the best accuracy using 512 keywords. The next step is to determine the best parameters. Table 4-10 shows the best 10-fold cross-validation accuracy of algorithms after trying different parameters with 512 features on the set C. For a list of applied parameters, refer to Table 4-1.

**Table 4-10: Best 10-fold cross-validation accuracy of decision trees on set C**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># of Trees</th>
<th># of features</th>
<th>Confidence</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5</td>
<td>1</td>
<td>-</td>
<td>0.1</td>
<td>77%</td>
</tr>
<tr>
<td>Random forest</td>
<td>15</td>
<td>10</td>
<td>-</td>
<td>80%</td>
</tr>
<tr>
<td>Random Tree</td>
<td>1</td>
<td>20</td>
<td>-</td>
<td>74%</td>
</tr>
</tbody>
</table>

The following table shows the 10-fold cross-validation accuracy of the algorithms when applied to sentences that contain only one pattern on set C by using different number of features. The keywords were selected by using information gain as explained in Section 3.3.6.

**Table 4-11: 10-fold cross-validation accuracy of algorithms after applying them on sentences with one pattern on set C**

<table>
<thead>
<tr>
<th># of keywords</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5</td>
<td>61%</td>
<td>70%</td>
<td>69%</td>
<td>71%</td>
<td>74%</td>
<td>73%</td>
<td>73%</td>
<td>73%</td>
<td>73%</td>
<td>73%</td>
</tr>
<tr>
<td>Random forest</td>
<td>62%</td>
<td>70%</td>
<td>70%</td>
<td>73%</td>
<td>75%</td>
<td>75%</td>
<td>77%</td>
<td>77%</td>
<td>76%</td>
<td>77%</td>
</tr>
<tr>
<td>Random Tree</td>
<td>63%</td>
<td>69%</td>
<td>66%</td>
<td>69%</td>
<td>70%</td>
<td>70%</td>
<td>71%</td>
<td>70%</td>
<td>70%</td>
<td>70%</td>
</tr>
</tbody>
</table>

We tried different parameters for random forest and random tree by using 512 features and different parameters for C4.5 by using 128 features on sentences with one pattern only. Table 4-12 shows the best 10-fold cross-validation accuracy on set C with the corresponding parameters. For a list of applied parameters, refer to Table 4-1.
4.3.3 SVM

We applied SVM by using four kernels: linear, polynomial, radial and sigmoid kernels. Firstly, we tried to determine the best number of keywords to consider as features. We used LIBSVM for training and testing an SVM classification model [88]. We used TF-IDF and z-score with DTFM. We performed 10-fold cross-validation. Table 4-13 shows the accuracy of each kernel after applying them on set C. The keywords were selected by using information gain as explained in Section 3.3.6.

<table>
<thead>
<tr>
<th># of keywords</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Kernel</td>
<td>63%</td>
<td>70%</td>
<td>70%</td>
<td>73%</td>
<td>76%</td>
<td>77%</td>
<td>72%</td>
<td>73%</td>
<td>72%</td>
<td>73%</td>
</tr>
<tr>
<td>Polynomial Kernel</td>
<td>63%</td>
<td>68%</td>
<td>67%</td>
<td>73%</td>
<td>72%</td>
<td>74%</td>
<td>71%</td>
<td>72%</td>
<td>69%</td>
<td>71%</td>
</tr>
<tr>
<td>Radial Kernel</td>
<td>63%</td>
<td>69%</td>
<td>72%</td>
<td>75%</td>
<td>76%</td>
<td>74%</td>
<td>70%</td>
<td>70%</td>
<td>69%</td>
<td>70%</td>
</tr>
<tr>
<td>Sigmoid Kernel</td>
<td>62%</td>
<td>67%</td>
<td>68%</td>
<td>71%</td>
<td>70%</td>
<td>73%</td>
<td>72%</td>
<td>70%</td>
<td>69%</td>
<td>70%</td>
</tr>
</tbody>
</table>

It is clear that 256 keywords is the best number of features. The next step is to determine the best parameters for each kernel. Table 4-14 shows the best 10-fold cross-validation accuracy with the corresponding parameters on set C. For a list of applied parameters, refer to Table 4-1.
Table 4-14: The best 10-fold cross-validation accuracy of SVM after applying it on set C

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Cost</th>
<th>Gamma</th>
<th>Coef0</th>
<th>Degree</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Kernel</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>77%</td>
</tr>
<tr>
<td>Polynomial Kernel</td>
<td>1</td>
<td>0.25</td>
<td>8</td>
<td>2</td>
<td>76%</td>
</tr>
<tr>
<td>Radial Kernel</td>
<td>1</td>
<td>0.5</td>
<td>-</td>
<td>-</td>
<td>69%</td>
</tr>
<tr>
<td>Sigmoid Kernel</td>
<td>1</td>
<td>0.125</td>
<td>8</td>
<td>-</td>
<td>62%</td>
</tr>
</tbody>
</table>

The 10-fold cross-validation accuracy of SVM after applying it on sentences that contain one pattern on set C only is shown in Table 4-15. The keywords were selected by using information gain as explained in Section 3.3.6.

Table 4-15: 10-fold cross-validation accuracy of SVM after applying it on sentences with one pattern only on set C

<table>
<thead>
<tr>
<th># of keywords</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Kernel</td>
<td>64%</td>
<td>69%</td>
<td>69%</td>
<td>73%</td>
<td>73%</td>
<td>69%</td>
<td>70%</td>
<td>70%</td>
<td>73%</td>
<td>72%</td>
</tr>
<tr>
<td>Polynomial Kernel</td>
<td>64%</td>
<td>69%</td>
<td>66%</td>
<td>71%</td>
<td>73%</td>
<td>70%</td>
<td>70%</td>
<td>69%</td>
<td>68%</td>
<td>69%</td>
</tr>
<tr>
<td>Radial Kernel</td>
<td>63%</td>
<td>69%</td>
<td>70%</td>
<td>73%</td>
<td>74%</td>
<td>70%</td>
<td>69%</td>
<td>69%</td>
<td>69%</td>
<td>69%</td>
</tr>
<tr>
<td>Sigmoid Kernel</td>
<td>64%</td>
<td>67%</td>
<td>67%</td>
<td>69%</td>
<td>70%</td>
<td>68%</td>
<td>70%</td>
<td>70%</td>
<td>72%</td>
<td>71%</td>
</tr>
</tbody>
</table>

All kernels give the best accuracy by using 128 features. Table 4-16 shows 10-fold cross-validation accuracy of SVM after applying it on sentences with one pattern on set C using 128 features and the best parameters. For a list of applied parameters, refer to Table 4-1.

Table 4-16: Best 10-fold cross-validation accuracy of SVM after applying it on sentences with one pattern on set C

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Cost</th>
<th>Gamma</th>
<th>Coef0</th>
<th>Degree</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Kernel</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>73%</td>
</tr>
<tr>
<td>Polynomial Kernel</td>
<td>1</td>
<td>0.0078</td>
<td>0</td>
<td>3</td>
<td>71%</td>
</tr>
<tr>
<td>Radial Kernel</td>
<td>1</td>
<td>0.0078</td>
<td>-</td>
<td>-</td>
<td>74%</td>
</tr>
<tr>
<td>Sigmoid Kernel</td>
<td>1</td>
<td>0.0078</td>
<td>0</td>
<td>-</td>
<td>72%</td>
</tr>
</tbody>
</table>
4.3.4 KNN Algorithm

We applied KNN [83] algorithm by using four distance measures: Euclidean, city block, cosine and correlation. Firstly, we tried to determine the best number of keywords as features. We used WEKA tool [87] for training and testing the model. We used TF-IDF and z-score with DTFM. We performed 10-fold cross-validation. Table 4-17 shows the accuracy after applying KNN on set C by using different number of features. The keywords were selected by using information gain as explained in Section 3.3.6.

<table>
<thead>
<tr>
<th># of keywords</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>61%</td>
<td>67%</td>
<td>68%</td>
<td>71%</td>
<td>70%</td>
<td>71%</td>
<td>69%</td>
<td>71%</td>
<td>70%</td>
<td>71%</td>
</tr>
<tr>
<td>City block</td>
<td>61%</td>
<td>68%</td>
<td>68%</td>
<td>72%</td>
<td>72%</td>
<td>73%</td>
<td>71%</td>
<td>70%</td>
<td>69%</td>
<td>70%</td>
</tr>
<tr>
<td>Cosine</td>
<td>61%</td>
<td>67%</td>
<td>68%</td>
<td>71%</td>
<td>69%</td>
<td>72%</td>
<td>69%</td>
<td>71%</td>
<td>68%</td>
<td>70%</td>
</tr>
<tr>
<td>Correlation</td>
<td>61%</td>
<td>68%</td>
<td>68%</td>
<td>71%</td>
<td>69%</td>
<td>72%</td>
<td>69%</td>
<td>71%</td>
<td>68%</td>
<td>70%</td>
</tr>
</tbody>
</table>

It is clear that KNN performs the best when using 256 features. Table 4-18 shows the 10-fold cross-validation accuracy of KNN after applying it on set C by using 256 features and a different number of neighbors.

<table>
<thead>
<tr>
<th># of neighbors</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>9</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>71%</td>
<td>67%</td>
<td>68%</td>
<td>66%</td>
<td>65%</td>
</tr>
<tr>
<td>City block</td>
<td>73%</td>
<td>68%</td>
<td>68%</td>
<td>67%</td>
<td>64%</td>
</tr>
<tr>
<td>Cosine</td>
<td>72%</td>
<td>68%</td>
<td>69%</td>
<td>67%</td>
<td>69%</td>
</tr>
<tr>
<td>Correlation</td>
<td>72%</td>
<td>68%</td>
<td>69%</td>
<td>67%</td>
<td>68%</td>
</tr>
</tbody>
</table>

The 10-fold cross-validation accuracy of KNN after applying it on sentences that contain only one pattern on set C is shown in the following table by using a different number of
keywords as features. The keywords were selected by using information gain as explained in Section 3.3.6.

Table 4-19: 10-fold cross-validation accuracy of KNN after applying it on sentences with one pattern on set C

<table>
<thead>
<tr>
<th># of keywords</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean Distance</td>
<td>61%</td>
<td>67%</td>
<td>66%</td>
<td>69%</td>
<td>70%</td>
<td>69%</td>
<td>69%</td>
<td>67%</td>
<td>67%</td>
<td>67%</td>
</tr>
<tr>
<td>City block Distance</td>
<td>61%</td>
<td>66%</td>
<td>68%</td>
<td>69%</td>
<td>70%</td>
<td>68%</td>
<td>70%</td>
<td>68%</td>
<td>68%</td>
<td>68%</td>
</tr>
<tr>
<td>Cosine Distance</td>
<td>61%</td>
<td>67%</td>
<td>66%</td>
<td>69%</td>
<td>71%</td>
<td>69%</td>
<td>69%</td>
<td>68%</td>
<td>68%</td>
<td>68%</td>
</tr>
<tr>
<td>Correlation Distance</td>
<td>61%</td>
<td>67%</td>
<td>65%</td>
<td>69%</td>
<td>70%</td>
<td>68%</td>
<td>69%</td>
<td>68%</td>
<td>67%</td>
<td>68%</td>
</tr>
</tbody>
</table>

It is clear that KNN performs the best by using 128 features. Table 4-20 shows the 10-fold cross-validation accuracy of KNN after applying it on sentences with one pattern on set C by using different neighbors and 128 features.

Table 4-20: 10-fold cross-validation accuracy of KNN by using different number of neighbors on sentences with one pattern on set C

<table>
<thead>
<tr>
<th># of neighbors</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>9</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean Distance</td>
<td>70%</td>
<td>67%</td>
<td>65%</td>
<td>67%</td>
<td>66%</td>
</tr>
<tr>
<td>City block Distance</td>
<td>70%</td>
<td>66%</td>
<td>66%</td>
<td>66%</td>
<td>65%</td>
</tr>
<tr>
<td>Cosine Distance</td>
<td>71%</td>
<td>65%</td>
<td>66%</td>
<td>67%</td>
<td>70%</td>
</tr>
<tr>
<td>Correlation Distance</td>
<td>70%</td>
<td>65%</td>
<td>66%</td>
<td>66%</td>
<td>69%</td>
</tr>
</tbody>
</table>

4.3.5 Naïve Bayes Algorithm

The last algorithm we applied is Naïve Bayes classification [85]. We used WEKA tool [87] for training and testing the model. We performed 10-fold cross-validation. We used frequency values with DTFM. The 10-fold cross-validation accuracy of Naïve Bayes with respect to different number of keywords after applying it on set C is in Table 4-21. The keywords were selected by using information gain as explained in Section 3.3.6.
Table 4-21: 10-fold cross-validation accuracy of Naïve Bayes after applying it on set C

<table>
<thead>
<tr>
<th># of keywords</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes</td>
<td>63%</td>
<td>69%</td>
<td>71%</td>
<td>75%</td>
<td>75%</td>
<td>77%</td>
<td>76%</td>
<td>77%</td>
<td>77%</td>
<td>77%</td>
</tr>
</tbody>
</table>

Table 4-22 shows the 10-fold cross-validation accuracy of Naïve Bayes when applied on sentences that contain only one pattern on set C. The keywords were selected by using information gain as explained in Section 3.3.6.

<table>
<thead>
<tr>
<th># of keywords</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes</td>
<td>65%</td>
<td>68%</td>
<td>70%</td>
<td>72%</td>
<td>72%</td>
<td>74%</td>
<td>75%</td>
<td>75%</td>
<td>75%</td>
<td>75%</td>
</tr>
</tbody>
</table>

4.4 Hybrid Approach

We used a hybrid approach that combined the two models that are based on the same machine learning algorithm. The first model is trained on sentences that contain one pattern only by using DTFM approach, while the second model is trained on sentences that contain multiple patterns by using PWMs approach. During the testing step if a sentence has only one pattern the first model that was trained using DTFM approach is used for classifying the sentence. Otherwise, if the sentence in the testing set contains more than one pattern then the second model that was trained using PWM approach is used. For example, if we want to use C4.5 then we will have two models of C4.5; one model is trained using DTFM on sentences with one pattern, and the second model is trained using PWMs on sentences with multiple patterns. Table 4-23 shows the 10-fold cross-validation accuracy of these algorithms by using the hybrid approach on set C.
Table 4-23: 10-fold cross-validation accuracy of different algorithms using the hybrid approach on set C

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>RE</th>
<th>F-score</th>
<th>ACC</th>
<th>PR</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random forest</td>
<td>86.27%</td>
<td>84.77%</td>
<td>83.50%</td>
<td>83.33%</td>
<td>80.35%</td>
</tr>
<tr>
<td>C4.5</td>
<td>80.35%</td>
<td>79.82%</td>
<td>78.38%</td>
<td>79.31%</td>
<td>76.13%</td>
</tr>
<tr>
<td>Random Tree</td>
<td>83.95%</td>
<td>83.57%</td>
<td>82.64%</td>
<td>83.57%</td>
<td>81.22%</td>
</tr>
<tr>
<td>SVM</td>
<td>70.37%</td>
<td>68.85%</td>
<td>67.41%</td>
<td>67.40%</td>
<td>64.45%</td>
</tr>
<tr>
<td>KNN</td>
<td>75.84%</td>
<td>73.05%</td>
<td>70.31%</td>
<td>70.46%</td>
<td>64.07%</td>
</tr>
</tbody>
</table>

4.5 Testing Results

We identified random forest with the hybrid approach as the best solution for the problem. Therefore, we evaluated its performance on the testing set T as in Table 4-24.

Table 4-24: Performance of the best algorithm on set T

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>RE</th>
<th>F-score</th>
<th>ACC</th>
<th>PR</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random forest – hybrid</td>
<td>99.00%</td>
<td>88.79%</td>
<td>87.50%</td>
<td>80.49%</td>
<td>76.00%</td>
</tr>
</tbody>
</table>

4.6 Summary of Results

This section summarizes the reported results in the previous sections. Figure 4-1 shows the performance of algorithms in 10-fold cross-validation after applying them on sentences with multiple patterns, sentences with one pattern only, and on the whole set C using PWMs approach. PWMs approach gives the best performance when applied on sentences with multiple patterns.
Figure 4-1: Summary of 10-fold cross-validation accuracy using PWMs approach on set C

Figure 4-2 shows the best 10-fold cross-validation accuracy of rule generation algorithms after applying them on sentences with one pattern only on set C and entire set C using DTFM approach.

Figure 4-2: Best 10-fold cross-validation accuracy of rule generation when applied to the entire set C and sentences with one pattern on set C
Figure 4-3 summarizes the 10-fold cross-validation accuracy of algorithms after applying them on set C using DTFM approach.

Figure 4-3: 10-fold cross-validation accuracy of algorithms using DTFM approach on set C

Figure 4-4 summarizes 10-fold cross-validation accuracy of algorithms after applying them on sentences with one pattern on set C using DTFM.

Figure 4-4: 10-fold cross-validation accuracy using DTFM approach on sentences with one pattern on set C
Figure 4-5 summarizes 10-fold cross-validation accuracy of Naïve Bayes on sentences with one pattern on set C and the entire set C.

![Accuracy Graph](image)

**Figure 4-5:** 10-fold cross-validation accuracy of Naïve Bayes on set C and sentences with one pattern on set C

Figure 4-6 summarizes the best 10-fold cross-validation accuracy of algorithms by using DTFM approach on the entire set C and on sentences with one pattern only on C.

![Algorithms Graph](image)

**Figure 4-6:** Best 10-fold cross-validation accuracy of all algorithms using DTFM approach on set C
Figure 4-7 compares the performance of algorithms when applied using DTFM and PWMs approaches on sentences with one pattern on set C and the entire set C.

Figure 4-7: Comparison of 10-fold cross-validation accuracy between DTFM and PWMs approaches on set C

Figure 4-8 compares the 10-fold cross-validation accuracy of hybrid, PWMs and DTFM approaches on set C.

Figure 4-8: Comparison between 10-fold cross-validation accuracy of Hybrid, PWM and DTFM approaches on set C
Chapter 5 Discussion

The aim of this chapter is to analyze the three approaches and algorithms in their performance in the problem we studied. Firstly, section 5.1 Comparison between PWMs, DTFM and Hybrid Approaches analyzes the performance of the three approaches. Secondly, section 5.2 Analysis of Algorithms’ Results analyzes the performance of the algorithms. Finally, section 5.3 Problems and Limitations describes some limitations of the system developed and the study.

5.1 Comparison between PWMs, DTFM and Hybrid Approaches

The classical approach to represent a document in text mining is by using DTFM to represent sentences. In this project, we introduced a different way for text representation by PWMs which is based on sentences analysis. Unlike DTFM, PWMs can keep track of the frequency of each word relative to predefined positions within a sentence. Then each sentence will be represented by a set of scores generated from each matrix.

Comparison 1

PWMs approach can represent a sentence in different ways depending on the contained pattern as explained in section 3.4.4. Therefore, this approach can be used to distinguish between positive and negative patterns even if they appear in the same sentence. On the other hand, DTFM represents a sentence of an abstract in one way regardless of the number of patterns that appear in the sentence. Therefore, this approach can be used to determine if a sentence has a positive pattern or not, but we cannot distinguish which pattern is the positive one and which one is negative, or how many positive pattern there are.
Comparison 2

DTFM approach generates a large number of features; hence there is an overhead of features selection step. However, PWMs generates a small number of features (only 12 features), so it is not required to perform the feature selection step.

Comparison 3

We applied PWMs approach by using sentences with one pattern only on set C, sentences with multiple patterns and the entire set C. PWMs approach gave better results when applied on set C or sentences with multiple patterns in set C as illustrated in Figure 4-1. We also found out that this approach gave worse results when applied on sentences with one pattern only on set C. Similarly, DTFM approach gave better results on the entire set C but worse when applied on sentences with one pattern only in set C as illustrated in Figure 4-6. One reason why both approaches did not perform the best on sentences with one pattern is because these sentences are relatively short. Since these sentences do not include enough words and thus enough information, it is not easy for classification algorithms to perform well.

Comparison 4

Five algorithms were applied by using DTFM and PWMs on set C (Random forest, C4.5, random tree, SVM, and KNN) as illustrated in Figure 4-7. The first three algorithms performed better by using PWMs approach. One reason why PWMs approach outperformed the classical document representation is that these scores can capture similarities between sentences better than DTFM approach.
Comparison 5
Similarly we applied the previous five algorithms by using DTFM and PWMs approaches on sentences with one pattern only on set C as illustrated in Figure 4-7. All the five algorithms performed better by using DTFM than PWMs.

Comparison 6
Each approach has strengths and weaknesses, and the previous analysis helped us understand the situations when each approach performed the best. PWMs approach gave the best results when applied to sentences with multiple patterns on set C. Similarly, DTFM gave better results than PWMs approach when applied on sentences with one pattern on set C. Therefore, a hybrid approach that can capture the strengths of both approaches and minimizes their weaknesses should be implemented.

The hybrid approach we implemented generates two models: one model trained on sentences with one pattern by using DTFM approach, and the other model trained on sentences with multiple patterns by using PWMs approach. The hybrid approach outperformed the PWMs and DTFM approaches when applied on set C with random forest, C4.5 and random tree as illustrated in Figure 4-8.

5.2 Analysis of Algorithms’ Results
In contrast to several text mining tasks, associations extraction is somewhat unique. Although positive and negative sentences have similar appearances, there are still some significant characteristics for each sentence class. Basically, a successful classification algorithm applied for this specific text mining domain must exploit possible variations
between sentence classes, be able to extract legitimate associations correctly, and produces only few false positives.

Similar to other applications of text mining, determining the best machine learning algorithm for associations extraction is a challenging task. However, the experiments we performed revealed several interesting features of classification algorithms. This section explains the characteristics of the classifiers that were applied and analyzes their performance.

5.2.1 Rule Generation Algorithms

Classification rules consist of patterns of words that best describe and represent a class of sentences. A new sentence is classified based on the rules that satisfy it. Overall, the main advantage of rule generation algorithms is that using patterns of words have more potential for classification decisions than using scores or similarity measures. The performance of any rule generation algorithm is based on generating a good set of classification rules. It can be very challenging to generate such set of rules if the underlying concept of the sentences is complex. However, even in such a situation, rule generation algorithms give a deep insight into the key predictive words of such concepts.

5.2.2 Decision Trees Algorithms

Decision trees are well-known classification algorithms. Each node in the tree represents a feature. Going down the tree, a decision must be made regarding to which direction to go at every node. At the leaf of the tree are the labels of the sentences. Decision trees can suffer from over-fitting (the tree has good performance on the training set but poor performance on the testing set). It is mainly caused by variance in the dataset. One way to
solve this problem is by using multiple trees. This is the main reason why random forest algorithm performed better than C4.5 or random tree when DTFM representation is used. However, all the three algorithms performed better when PWMs scores approach is used, since this approach reduced variance in the dataset.

5.2.3 SVM

SVM is a very stable algorithm and works very well with high dimensional data. It gave 77% accuracy by using DTFM. SVM try to separate two sets of sentences by a hyper-plane, which can increase the margin (the distance between the hyper-plane and the closest sentences). In this study, SVM is a very good classifier and performed very well compared to other classifiers using DTFM approach. The linear SVM is one of the best algorithms for text classification for our problem as indicated by the results in the previous chapter. SVM assumes that the data must be linearly separable in order to perform well, and possibly PWMs approach does not satisfy this assumption. This may be the main reason why SVM did not perform well with PWMs approach.

5.2.4 KNN

KNN depends on comparing the similarity between a new sentence and existing sentences. This algorithm selects the nearest $k$ sentences and classifies the new sentence by taking majority voting from the label of these $k$ sentences. KNN is a fast and efficient algorithm since it does not require any effort for training. Simply gather the training set and store it. The performance of KNN is mainly dependent on the value of $k$, and the similarity measure.
5.2.5 Naïve Bayes Algorithm

Naïve Bayes is a simple and a fast machine learning algorithm. It achieved 77% accuracy in the associations extraction task by using DTFM approach. It can perform very well when features are independent of each other. However, if some features exhibit dependencies, Naïve Bayes can perform poorly.

One possible explanation for the poor performance of Naïve Bayes when the number of features became large was the nature of scientific words. The language in scientific literature is ambiguous as explained in Chapter 1, and it is difficult to extract a set of features that lacks interdependency between keywords. All these observations lead to imprecise probability estimation and strong feature dependencies that can weaken the performance of Naïve Bayes.

5.3 Problems and Limitations

Problem & Limitation 1

Text mining tasks consist of several steps that form a data processing pipeline. The performance of last stages of the sequential tasks depends on the performance of the first stages. In our system that we developed to automatically extract associations between methylated genes and diseases, one of the main early steps is named entity recognition, which is responsible for extracting genes, diseases and methylation words. Entities such as genes and diseases could be very ambiguous, and it is very easy to generate many false positives because of wrongly identified named entities.
Problem & Limitation 2
Additionally, positive and negative sentences are very similar. Both categories of sentences must include the three concepts (genes, diseases and methylation words). However, there are very slight differences between sentences in the two categories, and it is a very challenging task to identify such differences.

Problem & Limitation 3
The fact there are more than one pattern in one sentence, and some sentences include positive and negative patterns at the same time makes the classification task very complicated. In this case, the automated system has to identify certain differences to determine what makes one pattern positive and the other one negative.

Problem & Limitation 4
The limited size of the dataset affected the training the testing steps, and the performance of the algorithms. The size of the dataset we used may not be enough to develop good training models.
Chapter 6 Conclusion

The objective of the study is to develop a method for extraction of associations between the methylated genes and disease and, based on these results, develop a system that is able to perform this task based on free text submitted by users. We achieved this goal, and we made several other contributions along the way. Yet, there is still an opportunity for improvement in the future work. This chapter summarizes findings and contributions of our study. Firstly, section 6.1 Summary of Findings summarizes the finding we obtained from the study. Section 6.2 Summary of Contributions summarizes the contributions we achieved. Then section 6.3 Recommendations discusses possible future work.

6.1 Summary of Findings

Finding 1
Random forest algorithm is a very competitive classifier, and it could stand out among all other algorithms we applied on data from this study. It outperformed all other algorithms in most of the cases using PWMs, DTFM, and hybrid approaches by achieving accuracy in 10-fold cross-validation of approximately 82%, 80% and 84%, respectively, and 88% of accuracy on the testing set T.

Finding 2
Based on our dataset, the hybrid approach appeared the best one to solve problem. It performed better than PWMs and DTFM approaches separately, because it combines strengths of both approaches as discussed in the previous chapter. The best accuracy achieved using the hybrid approach in 10-fold cross-validation is about 84% with random forest algorithm.
Finding 3

The performance of all algorithms is affected by the nature of sentences. All algorithms performed the worst when applied on sentences with one pattern. However, the best performance was when the algorithms were applied on sentences with multiple patterns. Also, these algorithms performed well when applied on entire set C, because set C includes sentences with multiple patterns as well as sentences with one pattern. If we apply these algorithms on sentences with multiple patterns only we can achieve accuracy of 86\% by using PWMs approach with random forest model.

6.2 Summary of Contributions

Contribution 1

One of the main contributions we made in this project is developing a new way for text representation in association extraction by using PWMs. Using PWMs has a great advantage since these matrices can be used to keep track of the frequency of words with respect to predefined positions. Also, this approach generates a small number of features, so it eliminates the feature selection step. We have shown in Chapter 4, that three algorithms (random forest, C4.5, and random tree) gave better performance by using PWMs approach.

Contribution 2

We developed a tool for automated extraction of associations between methylated genes and diseases from text submitted by users. We have conducted several experiments as shown in Chapter 4, and our results suggest that the hybrid approach is the best solution for the problem we studied. There is no available tool that can perform the same task, so we will be the first to offer this to research community in this field. Our approach does
not require setting any parameters or thresholds. The only input required from the user is the text that should be analyzed. The user can supply one sentence, one or more abstract or even one or more full-text documents. Then the system will process the input text, and will provide a list for methylated genes and diseases along with the sentences where they were found.

**Contribution 3**

The first module is intended for text preprocessing, but it can be used for automated sentence annotation. The module takes the sentences and the dictionaries of concepts as input, so it can be easily generalized to automate sentences with respect to any number of concepts. So for example, if we want to annotate sentences with respect to drugs and disease names, all we need to do is to supply the correct dictionaries and sentences, and the module will annotate the sentences.

**Contribution 4**

We have developed a module for automated generation of PWMs from text. It can be easily generalized to any number of concepts by supplying appropriate dictionaries. For example, we can generate PWMs with respect to four concepts (genes, disease, drugs and side effects), and then we can use these matrices to provide the features. Later these features can be used for training and testing any classification model developed for such data representation.

**Contribution 5**

We have conducted a comparative study of performances between twelve different algorithms. Such study is very important for better understanding of associations extraction task in text mining.
Contribution 6

We have classified a dataset manually. Considering the limited number of available datasets for biomedical literature, such dataset can be very useful for training and testing certain types of text mining algorithms.

6.3 Recommendations

Recommendation 1

In order to improve the performance of our system to extract associations between methylated genes and diseases, we can integrate our system with rule generation. We can generate a set of rules to describe misclassified sentences, and these rules can be used during the classification step along with the hybrid approach.

Recommendation 2

We also plan to develop a toolbox for automated PWMs generation for text. This toolbox can be a toolbox for MATLAB, a library for Python or any other programming language. People will be able to use the toolbox to generate PWMs, normalize the matrices, and then using the matrices to compute the scores and generate the features. This tool will be very useful for people who intend to use PWMs approach for associations extraction.

Recommendation 3

Additionally we propose to apply our system on the entire PubMed database to extract all associations between methylated genes and diseases. We will be able to get a list of associations that appear in these abstracts, which may not exist in current methylation databases. Such discovered associations can be of great value for scientists who work in genes methylation domain.
BIBLIOGRAPHY


87


58. Han, J. and M. Kamber, Data mining: concepts and techniques 2006: Morgan Kaufmann.


69. Coenen, F., LUCS KDD implementation of CBA (Classification Based on Associations). 2004.

70. Li, W., J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. 2001. IEEE.


