CHAPTER 2

INFORMATION EXTRACTION BACKGROUND

Information Extraction (IE) [16] is any process which selectively structures and combines data which is found, explicitly stated or implied, in one or more texts. It is a part of text mining that refers to the process of extracting patterns or knowledge from unstructured text documents. It can be viewed as an extension of data mining or knowledge discovery [17]. The most natural form of storing information is in text where each text element depends on the topic fields. This thesis focuses on the biological extraction of textual information from literature which is related to specific functional genomics. **Chapter 1** explored how biological terms are very special and have a different form from other words. Hence, the preliminary of bio text mining and techniques which are used for information extraction is explained in the next section.

2.1 Introduction

The large volume of biological or biomedical literature and its continuing fast growth has created an increasingly important need for text mining tools. Text mining and information-extraction approaches have been developed to extract relevant information such as proteins, and DNA Analysis with the aim of helping biologists to transform available data into usable information and knowledge.

In biology, the information resources available are, essentially, a vast collection of databases that cover a broad range of source types such as keywords, protein sequences,

abstracts and structural information. In addition, some databases focus on specific aspects of protein function. The primary source of free textual data information in molecular biology and biomedicine is Medline, which is a collection of more than 12,000,000 abstracts maintained by the National Library of Medicine (NLM) that is commonly accessed by biologists using the PubMed suite as shown in **Figure 2.1**

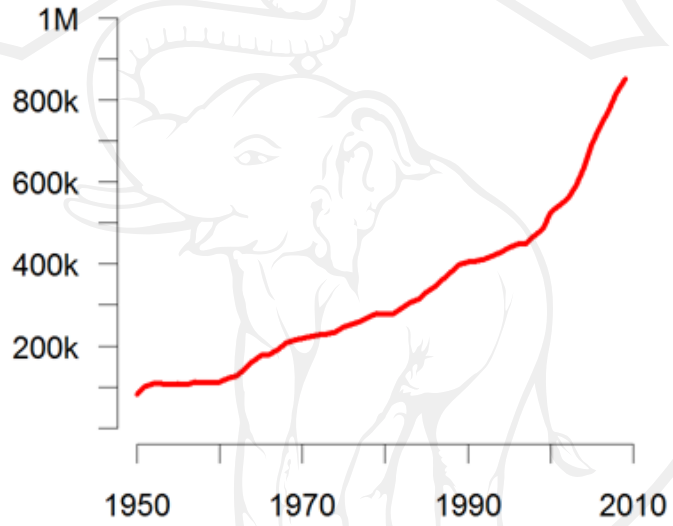

Figure 2.1 Number of MEDLINE-indexed articles published per year

In general, text mining applications take advantage of a range of domain-independent methods such as part of speech (POS) taggers, which label each word with its corresponding part of speech (e.g. noun, verb or adjective), or stemmers, which are algorithms that return the morphological root of a word form. Also, domain specific tools and resources such as protein taggers and ontologies are employed. Information extraction attempts to identify biologically meaningful semantic structures within free text using strategies based on POS information, ontologies or the identification of

patterns. An example of the use of information extraction applications in molecular biology is the identification of protein interactions.

In the biological domain, extracted entities often correspond to proteins, genes, diseases or chemical compounds, for which automated identification methods are often incorporated. For the extraction of entities, parsing tools and POS taggers that can detect verbs of interest are also often useful. The pattern matching and syntactic analysis techniques can also highlight relevant text passages from large abstract collections.

For the analysis of biological literature using NLP, the field of NLP is concerned with the analysis of free textual information and has been applied recently in the context of molecular biology. Biological text mining approaches also involve analyzing and extracting information from large collections of free textual data by using automatic or semiautomatic systems. Currently, text mining applications are being employed in the identification of biological entities such as protein or gene names, automated protein annotation, analysis of microarrays and extraction of protein–protein interactions.

However, generating new insights to direct future research is far more complex. The goal of knowledge discovery is to find hidden information in the literature by exploring the internal structure of the knowledge network created by the textual information. Knowledge discovery could be of major help in the discovery of indirect relationships, which might imply new scientific discoveries. Such new discoveries might provide hints for experts working on specific biological processes.

There are many applications relates with biological text mining such as information retrieval of biological articles, DNA expression arrays, functional annotation and tagging

biological entities. In biomedical literature, the identification of biological entities such as gene and protein names, chemical compounds and diseases is crucial for facilitating the retrieval of relevant documents and the identification of relationships between those biological entities (e.g. between proteins and diseases). Most TM systems have not relied on ontologies or terminologies, which is a main reason why biomedical TM systems generally provide poorer results compared to other domains (e.g. newswire) [18].

Biological language and vocabulary is highly complex and rapidly evolving, making the identification of entities a cumbersome task, especially in the case of protein and gene names. When labeling text relative to the occurrence of genes or proteins, several obstacles are encountered. First, a variety of alternative expressions can occur that refer to the same protein object are often encountered; proteins might be mentioned in documents in terms of gene symbols, protein names, synonymous gene names and typographical variants. Moreover, some gene symbols are ambiguous and might correspond to disease names or experimental methods. These are all reasons why bio text mining is still a challenging task [4].

2.2 Effectiveness of General Term in Biological Literature

2.2.1 Introduction

From the last section, terms or words are referred to be the main reason of the different from other topics. In this section, text analysis is studied. One of the text analysis methods is a tokenization technique which is the process of extracting plain words and terms from a document and stripping out administrative metadata and

structural or formatting elements. This operation needs to be performed prior to indexing or before converting documents to vector representations that are used for retrieval or categorization. Tokenization appears to be a straightforward problem but in many practical situations the task can actually be quite challenging. The simplest approach consists of reducing the document to an unstructured representation such as a plain sequence of words with no particular relationship among them other than a serial order, removing tags, and perhaps converting strings that encode international characters to a standard representation. The resulting unstructured representation allows simple queries to be run related to the presence of terms and to their positional vicinity.

After plain text is extracted, punctuation and other special characters need to be stripped off. In addition the character case may be folded to reduce the number of index terms. However these strings are not necessarily entire words. Words are typically split into fragments and separate show commands are issued for each of the fragments. Hence, it is necessary to track the position of each string and use information about the font in order to correctly reconstruct word boundaries.

2.2.2 Methods and Dataset

To study the effectiveness of general term which appear in biological literature, the experiment framework is designed as shown in **Figure 2.2**

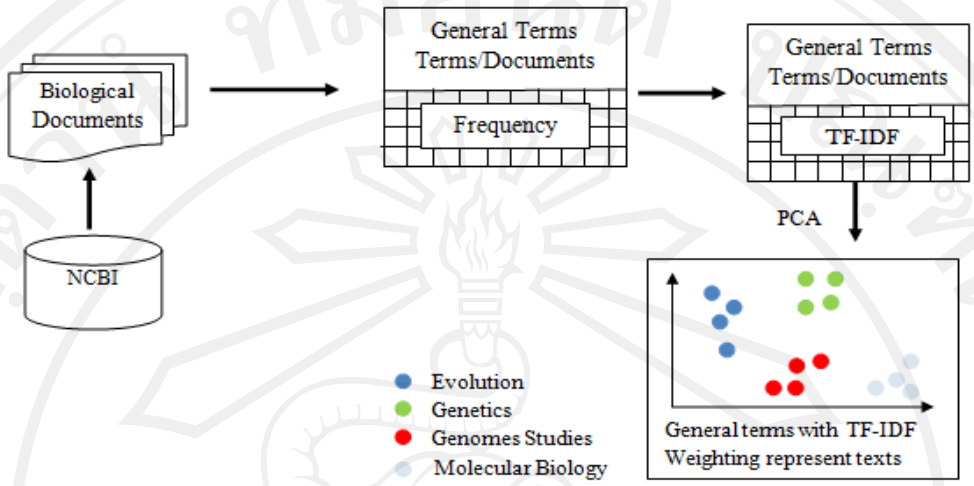

Figure 2.2 The framework of effectiveness of general term in biological literature

From **Figure 2.2**, the biological documents from NCBI are collected. Then, general terms are counted and transformed to the matrix of general terms /documents. After that, the matrix will scale with TF-IDF and visualized by Principle Component Analysis (PCA). The graph in **Figure 2.2** shows the expectation of how general terms could represent its class. The methods used in this experiment as follow:

1) Term-document matrix scaling by TF-IDF

For the case of two sparse vectors x and y associated with two documents A and A', the above sum can be computed efficiently in time $\Omega(|A| + |A'|)$. Several refinements can be obtained by extending the Boolean vector model and introducing real valued weights associated with terms in a document. A more informative weighting scheme consists of counting the actual number of occurrences of each term in the document. In this case $x_j \in N$ counts term occurrences in the corresponding document. x may be multiplied by the constant $1/|A|$ to obtain a vector of term frequencies (TF) within the document. An important family of weighting schemes combines term frequencies (which are relative to each document) with an absolute measure of term importance called

inverse document frequency (IDF). IDF importance decreases as the number of documents in which the term occurs increases in a given collection. Hence, terms that are globally rare receive a higher weight. Formally, let $D = \{A_1, \ldots, A_n\}$ be a collection of documents and for each term ω_j let n_{ij} denote the number of occurrences of ω_j in A_i and n_j the number of documents that contain ω_j at least once.

Define,

$$IDF_j = log \frac{n_j}{n} \tag{2.1}$$

$$TF_{ij} = \frac{n_{ij}}{|d_i|} \tag{2.2}$$

Here the logarithmic function is employed as a damping factor. The TF–IDF weight of ω_j in d_i can be computed as

$$x_{ij} = TF_{ij} \cdot IDF_j \tag{2.3}$$

The IDF weighing is commonly used as an effective heuristic. A theoretical justification has been recently proposed by Papineni (2001) [19].

2) General terms visualization by Principle Component Analysis (PCA)

PCA is a technique that is useful for the compression and classification of data. The purpose is to reduce the dimensionality of a data set (sample) by finding a new set of variables, smaller than the original set of variables, which nonetheless retains most of the sample's information. By information we mean the variation present in the sample, given by the correlations between the original variables. The new variables, called principal components (PCs), are uncorrelated, and are ordered by the fraction of the total

information each retains. The other main advantage of PCA is that once you have found these patterns in the data, you can compress the data, by reducing the number of dimensions, without much loss of information. The PCA is computed by determining the eigenvectors and eigenvalues of the covariance matrix. The covariance of two random variables is their tendency to vary together [20]. This is expressed as:

$$cov(X,Y) = E[E[X] - X] \cdot E[E[Y] - Y] \tag{2.4}$$

where E[X] denotes the expected value of X. For sampled data this can be explicitly written out as:

$$cov(X,Y) = \sum_{i=1}^{N} \frac{(x_i - \bar{x})(y_i - \bar{y})}{N} \tag{2.5}$$

With $\bar{x}$ = mean(X) and $\bar{y}$ = mean(Y) cov(X, X) = var(X), and for independent variables cov(X, Y) = 0. The covariance matrix is a matrix A with elements $A_{i,j}$ = cov(i, j). The covariance matrix is square and symmetric. For independent variables, the covariance matrix will be a diagonal matrix with the variances along the diagonal. To calculate the covariance matrix from a dataset, first center the data by subtracting the mean of each sample vector. Considering the columns of the data matrix A as the sample vectors, we can write the elements of the covariance matrix C as:

$$c_{ij} = \frac{1}{N} \sum_{i=1}^{N} a_{ij}\, a_{ji} \tag{2.6}$$

written in matrix form:

$$C = \frac{1}{N} AA^T \quad (2.7)$$

Often the scale factor 1/N is distributed throughout the matrix and the covariance matrix is written simply as AA^T . The eigenvectors of the covariance matrix are the axes of maximum variance. The PCA technique is widely used because of the fact that in many datasets, a majority of the variance of the data can be captured by a small subset of the eigenvectors.

3) Dataset

120 abstracts from the NCBI were used as the dataset. They are 4 classes [Evolution, Genetics, Genome Studied and Molecular Biology].The corpus is in the XML platform format**.**

2.2.3 Results and discussion

Term-document matrix were scaled by IF-IDF, (equations 2.3) as is shown in **Figure 2.3**

Term/Doc	gb-2002-3-	gb-2002-3-	gb-2002-3-	gb-2002-3-	gb-2002-3-	gb-2002-3-	gb-2002-3-	gb-2002-3-	gb-2002-3-	gb-2002-3-	gb-2004-5-	gb-2004-5-	gb-2005-6-	gb-2005-6
evalu	0.698	0	0	0	0	0	0	0	0	0	0	0	0	0
threshold	1	0	0	0	0	0	0	0	0	0	0	0	0	0
detect	0.145	0	0	0	0	0	0	0	0	0	0	0	0	0
bind	0.207	0	0	0	0	0	0	0	0	0	0	0	0	0
site	1	0	0	0	0	0	0	0	0	0	0	0	0	0
regulatori	1	0	0	0	0	0	0	0	0	0	0	0	0	0
protein	0.154	0.154	0	0	0	0	0	0.309	0	0.154	0	0	0	0
escherichi	1	0	0	0	0	0	0	0	0	0	0	0	0	0
coli	0.256	0	0	0	0	0	0	0	0	0	0	0	0	0
k12	1	0	0	0	0	0	0	0	0	0	0	0	0	0
dna	0.308	0	0	0	0	0	1	0	0	0	0	0	0	0
molecular	0	1	0	0	0	0	0	0	0	0	0	0	0	0
genet	0	0.076	0	0	0	0	0	0	0.076	1	0	0	0	0
structur	0	1	0	0	0	0	0	0	0	0	0	0	0	0
genom	0	1	1	0	0	0	1	0	0	0	0	0	0	0
human	0	1	0	0	0	0	0	0	1	1	0	0	0	0
kinas	0	1	0	0	0	0	0	0	0	0	0	0	0	0
c	1	0	1	0	0	0	1	0	0	0	0	0	0	1
gene	0	0.124	0	0.236	1	0.124	1	0	1	0.178	0	1	0	0
modul	0	1	0	1	0	0	0	0	0	0	0	0	0	0
evolutionar	0	0	1	0	0	0	0	0	0	0	0	0	0	0
conserv	0	0	1	0	0	0	0	1	0	0	0	0	0	0
otd/otx2	0	0	1	0	0	0	0	0	0	0	0	0	0	0
transcript	0	0	1	0	0	0	0	0	0	0	0	0	0	0
factor	0	0	1	0	0	0	0	0	0	0	0	0	0	0
action	0	0	1	0	0	0	0	0	0	0	0	0	0	0
genomwid	0	0	0.567	0	0	0	0	0	0	0	0	0	0	0
microarrai	0	0	1	0	0	0	1	0	0	0	0	0	0	0
analysi	0	0	1	0	0	0	0	0	0	1	0	0	0	0
drosophila	0	0	1	1	0	0	0	0	0	0	0	0	0	0

Figure 2.3 Term-document matrix scaling by TF-IDF

TF-IDF is mentioned that can find documents that make frequent use of said words and determine if they are relevant in the document. Then to study the effectiveness of general term in biological literature, the matrix with TF-IDF is visualized by PCA. The results is shown in **Figure 2.4**,

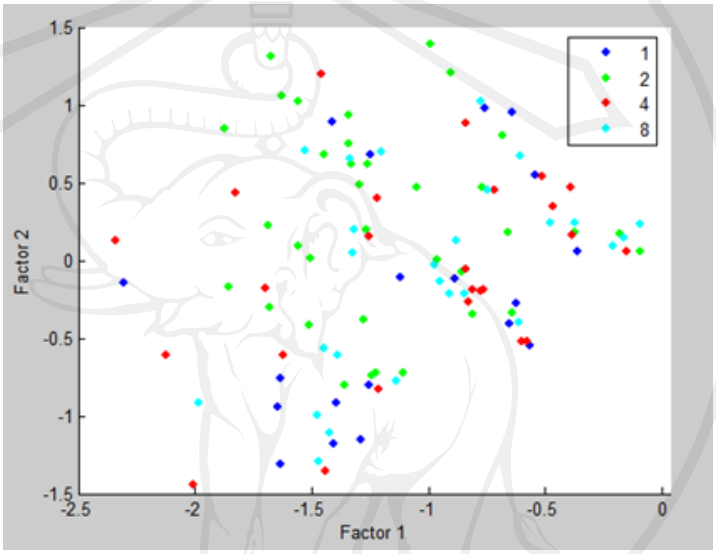

Figure 2.4 Abstracts classification with general term by TF-IDF Scaling

From **Figure 2.4**, the results showed that general terms could not represent text or document for each class well. In a biological content, many terms appear in every class. Hence, general terms were not enough for biological information extraction.

2.3 Biological Information Retrieval using Latent Semantic Indexing

2.3.1 Introduction

The information in the literature and documents is currently increasing. In order to extract knowledge, information retrieval of all relevant documents is essential. The

ultimate goal of information retrieval and extraction is the automatic transformation of unstructured textual information into a structured form [21]. According to information extraction is a sub-task of information retrieval where the goal is to extract structured information from unstructured text, section 2.2 focused on the effectiveness of retrieval of general terms in the biological literature. This section aimed to study the effectiveness of general terms to the biological information retrieval system**.**

2.3.2 Methods and Dataset

The experiment framework is designed. After scaling by TF-IDF, eigenvectors are generated with Latent Semantic Indexing (LSI). Then information retrieval performance is tested with query vector by the Cosine Similarity technique as **Figure 2.5**.

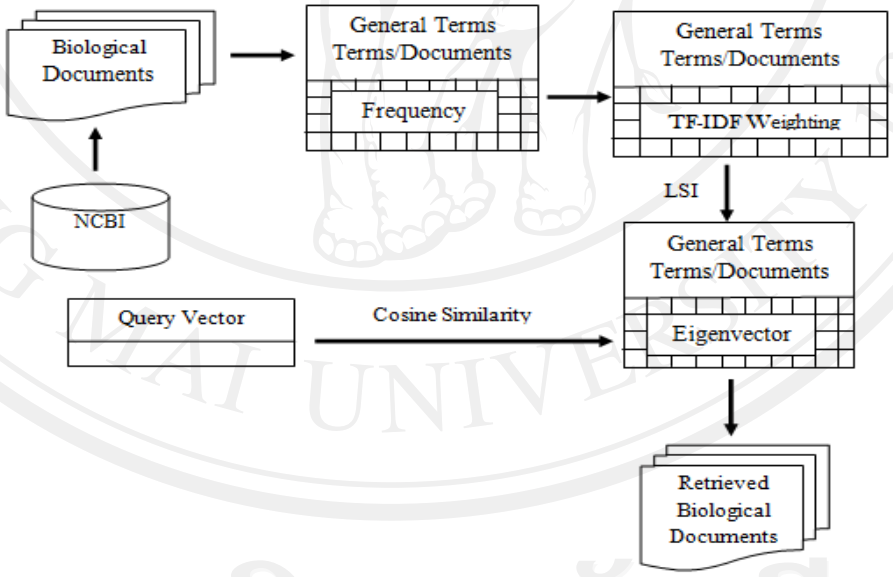

Figure 2.5 The framework of biological information retrieval using LSI

The methods which used in this section were LSI and Cosine Similarity as follow,

1) Latent Semantic Indexing (LSI)

This retrieval approach is based on vector space similarities that can reach satisfactory recall rates only if the terms in the query are actually present in the relevant documents. Natural language has a very rich expressive power and even at the lexical level, the large variability due to synonymy and polysemy can cause serious problems for retrieval methods based on term matching. Synonymy means that the same concept can be expressed using different sets of terms. Synonymy negatively affects recall. Polysemy means that identical terms can be used in very different semantic contexts. Polysemy negatively affects precision. Overall, synonymy and polysemy lead to a complex relation between terms and concepts that cannot be captured through simple matching.

Thus, although a query may conceptually be very close to a given set of documents, its associated vector could be orthogonal or nearly orthogonal to those document vectors, simply because the authors of the document and the user have a different usage of language. Another more statistical way of thinking about this is to observe that the number of terms that are present in a document is a rather small fraction of the entire dictionary, reducing the likelihood that two documents use the same set of words to express the same concept.

Latent Semantic Indexing (LSI) is a statistical technique that attempts to estimate the hidden structure that generates terms for given concepts. It uses a linear algebra technique known as Singular Value Decomposition (SVD) to discover the most important associative patterns between words and concepts. LSI is a data driven method, where a

large collection of sentences or documents is employed to discover the statistically most significant co-occurrences of terms.

- LSI and text documents

Let $\boldsymbol{X}$ denote a term–document matrix, defined as

$$X = [x_1 \cdots x_n]^T \tag{2.8}$$

Each row in the matrix is simply the vector-space representation of a document. In LSI each column contains the occurrences of a term in each document in the data set. The LSI technique consists of computing the SVD of $\boldsymbol{X}$ and setting to zero all the singular values except the largest *K* ones. In practical applications *K* is often set to values between 100 and 1000. This can be seen as analogous to the PCA dimensionality reduction: documents are mapped to a lower-dimensional latent semantic space induced by selecting directions of maximum covariance. The reconstructed matrix $\widehat{\boldsymbol{X}}$ is not as sparse as the original matrix $\boldsymbol{X}$. The new term weights account for word co-occurrences and appear to infer relations among words that pertain to synonymy, at least in a loose sense. When performing the numerical computation of the SVD of a very large document matrix, it is important to take advantage of sparsity. A variety of well-known numerical methods for computing the SVD of sparse matrices can be brought to bear (Berry and Browne 1999) [22].

Latent semantic indexing (LSI) is a statistical technique that attempts to estimate the hidden structure that generates terms given concepts [22]. It uses a linear algebra technique known as singular value decomposition (SVD) to discover the most important associative patterns between words and concepts. LSI is a data-driven method, where a

large collection of sentences or documents is employed to discover the statistically most significant co-occurrences of terms.

Let $\boldsymbol{X}$ denote a term–document matrix, defined as $X = [x_1, x_2, \ldots, x_n]^T$, where each row in the matrix is simply the vector-space representation of a document. In LSI, use the occurrence counts as components. Each column contains the occurrences of a term in each document in the data set. The LSI technique consists of computing the SVD of $\boldsymbol{X}$ and setting to zero all the singular values except the largest *k* ones (*k* is the dimension). Documents are mapped to a lower dimensional latent semantic space induced by selecting directions of maximum covariance. Performing SVD in this example yields a singular value Matrix. The reconstruction of $\boldsymbol{X}$ is obtained as $\boldsymbol{X}^{\wedge} = \boldsymbol{US}^{\wedge}\boldsymbol{V}^{\mathrm{T}}$. The new term weights account for word co-occurrences and appear to infer relations among words that pertain to synonymy, at least in a loose sense. SVD has inferred a link between features through the co-occurrences of other words (they never occur together).When performing the numerical computation of the SVD of a very large document matrix, it is important to take advantage of sparsely. A variety of well-known numerical methods for computing the SVD of sparse matrices can be brought to bear.

For the LSI process, words in the document are transformed into a vector form to construct the term-document and query matrices, then decompose the matrix term-document matrix X as $\mathbf{X = USV}^{\mathrm{T}}$ to rank documents in a decreasing order of query-document cosine similarities [23]. The Similarity is measured by the dot product between the query and document vector coordinates divided by the product of the norms of the query and document vectors**.**

2) Cosine Similarity

A Boolean query to a search engine may return several thousand matching documents, but a typical user will only be able to examine a small fraction of these. Ranking matching documents according to their relevance to the user is therefore a fundamental problem. In this section will review some classics models.

- **The vector space model and document similarity**

Text documents can be conveniently represented in a high dimensional vector space where terms are associated with vector components. More precisely, a text document d can be represented as a sequence of terms, $d = (\omega(1), \omega(2), \ldots, \omega(|d|))$, where $|d|$ is the length of the document and $\omega(t) \in V$. A vector-space representation of d is then defined as a real vector $\boldsymbol{x} \in R^{|V|}$, where each component x_j is a statistic related to the occurrence of the j^{th} vocabulary entry in the document. The simplest vector based representation is Boolean, i.e. $x_j \in \{0, 1\}$ indicates the presence or the absence of term ω_j in the document being represented. Vector based representations are sometimes referred to as a bag-of-words, emphasizing that document vectors are invariant with respect to term permutations, since the original word order $\omega(1), \ldots, \omega(|v|)$ is clearly lost. Representations of this kind are appealing for their simplicity. Moreover, although they are necessarily lossy from an information theoretic point of view, many text retrieval and categorization tasks can be performed quite well in practice using the vector space model. The total number of terms in a set of documents is much larger than the number of distinct terms in any single document, $|V| >> |d|$, so that vector space representations tend

to be very *sparse*. This property can be advantageously exploited for both memory storage and algorithm design.

Two documents *A* and *A*' are defined similarity as a function *s(d, d')* ∈ R. This function allows ranking documents with respect to a query (by measuring the similarity between each document and the query). A classic approach is based on the vector space representation and the metric defined by the *cosine coefficient* (Salton and McGill 1983) [24]. This measure is simply the cosine of the angle formed by the vector-space representations of the two documents, ***x*** and ***x***' (see **Figure 2.6)**

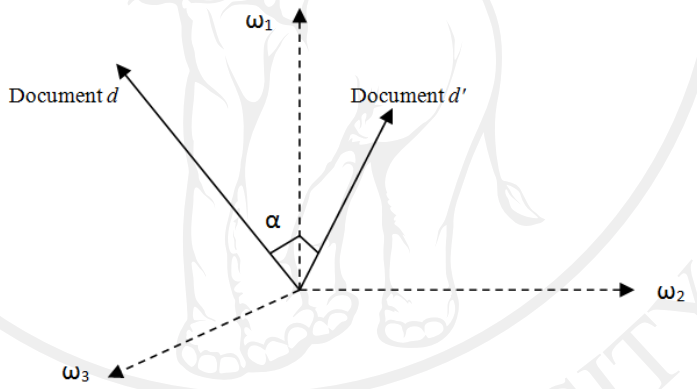

Figure 2.6 Cosine measure of document similarity

$$cos(x, x') = \frac{x^T x'}{\|x\| \cdot \|x'\|} = \frac{\sqrt{x^T x'}}{\sqrt{x^T x'} \cdot \sqrt{x^T x'}} \tag{2.9}$$

where the superscript T denotes the transpose operator and $x^T y$ indicates the dot product or inner product between two vectors $x, y \in R^m$, defined as

$$x^T y = \sum_{i=1}^{m} x_i y_i \qquad (2.10)$$

The case of two sparse vectors *x* and *y* associated with two documents *A* and *A*', the above sum can be computed efficiently in time *Ω (|A| + |A'|)*. Several refinements can be obtained by extending the Boolean vector model and introducing real valued weights associated with terms in a document.

3) Dataset

120 abstracts from NCBI were used as the dataset. They were 4 classes [Evolution, Genetics, Genome Studied and Molecular Biology].The corpus is in the XML platform format. (The same as section 2.3)

2.3.3 Results and Discussion

In the validation step, the performance of this work was evaluated by precision and recall values. Precision π (equation 2.11) is defined as the fraction of retrieved documents that are actually relevant. Recall ρ (equation 2.12) is defined as the fraction of relevant documents that are retrieved by the system. Then, harmonic mean of precision and recall are weighted with the traditional F-measure. The concept is as follow,

Let's consider a collection of *n* documents *D*. Each document is represented by an *m*-dimensional vector, where $m = |V|$ and *V* is the set of terms that occurred in the collection. Let $\boldsymbol{q} \in R_m$ denote the vector associated with a user query (terms that are present in the query but not in *V* will be stripped off). Each document is then assigned a score, relative to the query, by computing $s(\boldsymbol{x}_i, \boldsymbol{q})$, $i = 1, \ldots, n$. The set *R* of *retrieved*

documents that are presented to the user can be formed by collecting the top ranking documents according to the similarity measure. The quality of the returned collection can be defined by comparing R to the set of documents R^* that is actually relevant to the query. Two common metrics for comparing R and R^* are *precision* and *recall*. Precision π is defined as the fraction of retrieved documents that are actually relevant. Recall that ρ is defined as the fraction of relevant documents that are retrieved by the system. More precisely,

$$\pi = \frac{|R \cap R^*|}{|R|} \tag{2.11}$$

$$\rho = \frac{|R \cap R^*|}{|R^*|} \tag{2.12}$$

In this context the ratio between relevant and irrelevant documents is typically very small. For this reason, other common evaluation measures like accuracy or error, where the denominator consists of $|D|$. Sometimes precision and recall are combined into a single number called F_β measure defined as

$$F_\beta = \frac{(\beta^2 + 1)\pi\rho}{\beta^2\pi + \rho} \tag{2.13}$$

The $F1$ measure is the harmonic mean of precision and recall. If β tends to zero (∞) the F_β measure tends to precision (recall).

The results of experiment showed in Table 2.1

Table 2.1 Information retrieval model performance which generated by general terms and LSI

	Recall (%)	**Precision (%)**	**F-Measure (%)**
General Terms + LSI	65.00	67.00	65.98

The result from **Table 2.1**, the model returned precision value higher than recall by 2%. This shows that the model returns fewer false positives than false negatives. LSI helped to discover the most important associative patterns between words and concepts. But the F-measure returns only almost 66%. This showed that the general terms still were not enough for a general information retrieval system.

2.4 Biological Text Classification using Machine Learning Techniques

2.4.1 Introduction

Biological Information extraction can relate to many techniques. The section above presented general terms which were used for factor in machine learning not only in information extraction but also in information retrieval. The results showed that general terms still could not represent its class well. Hence, this work focused on generating general terms as features. The feature selections based on machine learning were studied and used to generate classification models.

2.4.2 Methods and Dataset

The framework of this work was divided into two processes. The first was feature selection based on machine learning. The matrix scaling by TF-IDF was selected to be

the feature for the feature selection models. This work focused on traditional techniques such as Information Gain, Mutual Information and Odd Ratio while another was classification models. The *k*-nearest neighbor, Naïve Byes and Support Vector Machines were the scope which studied as **Figure 2.7,**

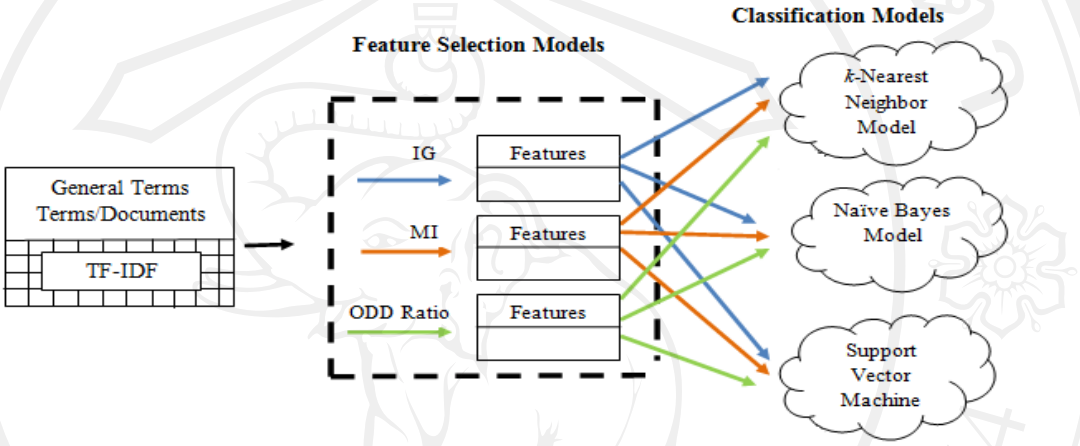

Figure 2.7 The framework of bioinformatics-text classification using machine learning techniques

Three of the most important and effective machine learning algorithms that are often applied to text classification: *k*-nearest neighbors (*k*-NN), Naive Bayes, and Support Vector Machines (SVMs) are briefly reviewed**.**

1) Text Classification models

- *k*-Nearest Neighbors

k-NN is a memory based classifier that learns by simply storing all the training instances. During prediction, the *k*-NN algorithm first measures the distances between a new point ***x*** and all the training instances, returning the set *N(**x**, D, k)* of the *k* points that are closest to ***x***. For example, if training instances are represented by real-valued vectors

x, the Euclidean distance is used to measure the distance between ***x*** and all other points in the training data, i.e.

$$||\boldsymbol{x}-\boldsymbol{x}_i||^2 \tag{2.14}$$

where $i = 1, \ldots, n$.

After calculating the distances, the algorithm predicts a class label for ***x*** by a simple majority voting rule using the labels in the elements of *N(**x**, D, k)*, breaking ties arbitrarily. In spite of its apparent simplicity, *k*-NN is known to perform well in many domains. The results by Cover and Hart (1967) [25] show that the asymptotic error rate of the 1-NN classifier (as the size of the training data set gets infinitely large) is always less than twice the optimal Bayes error (which is the lowest possible error rate achievable by *any* classifier in a particular feature space ***x***). In the case of texts, the majority voting can be replaced by a smoother metric where, for each class *c*,

$$s(c|x) = \sum_{x' \in N_c(x,D,k)} \cos(x, x') \tag{2.15}$$

a scoring function is computed through vector-space similarities between the new documents and the subset of the *k* neighbors that belong to class *c*, where *Nc(**x**, D, k)* is the subset of *N(**x**, D, k)* containing only points of class *c*. Despite the simplicity of the method, the performance of *k*-NN in text categorization is quite often satisfactory in practice. Han *et al.* (2001) [26] have proposed a variant of *k*-NN where the weights associated with features are learned iteratively while the other statistically motivated

techniques that extend the basic *k*-NN classifier are also discussed in Hastie *et al.* (2001) [27]

- Naïve Bayes

This classifier attempts to estimate the conditional probability of the class given the document, namely *P(c | d)*, for *c* = 1, . . . , *S*. Using Bayes' theorem, it can write this probability as

$$P(c|d,\boldsymbol{\theta}) = \frac{P(d|c,\boldsymbol{\theta})P(c|\boldsymbol{\theta})}{P(d|d,\boldsymbol{\theta})} \alpha\, P(d|c,\boldsymbol{\theta})P(c|\boldsymbol{\theta}) \qquad (2.16)$$

where $\boldsymbol{\theta}$ are the parameters of the model. The classes are assumed to be mutually exclusive, the term that can be thought of as a normalization factor guarantees that $\boldsymbol{\Sigma}$ *c P(c | d)* = 1. The key idea behind the Naive Bayes classifier is the assumption that the terms in a document are conditionally independent given the class. This assumption is clearly false in many if not most practical situations, but it is often adequate to first order for the bag of words representation, where word order in the document is not taken into account. Ng and Jordan (2002) [28] mention that this should discriminate among classes, not in a high quality generative model of the document given the class. In practice, the classifier is known to work satisfactorily even when the conditional independence assumption is known not to hold (Domingos and Pazzani 1997) [29]

There is a subtle issue concerning the interpretation of the document in terms of a probabilistic event (McCallum and Nigam 1998) [30]. If the document as a whole is considered to be an event, then it should be naturally described by a bag of words, and the words are the *attributes* of this event. In this case, each vocabulary term is associated

with a Bernoulli attribute whose realization is unity if the term appears in the document, and zero otherwise. In addition to reducing documents to bags of words, the Naive Bayes model postulates that binary attributes are mutually independent given the class.

The conditional independence assumption in this model can be depicted graphically using a Bayesian network, suggesting that the class is the only cause of the appearance of each word in a document.

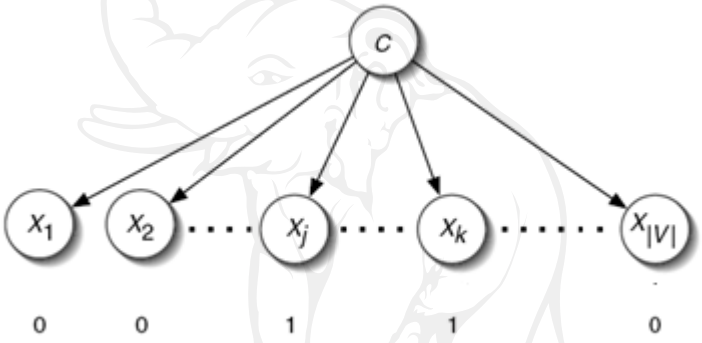

Figure 2.8 A Bayesian network for the Naive Bayes classifier under the Bernoulli document-based event model.

The example document is the sentence in which appears words x_j and x_k. Under this model, generating a document is like tossing $|V|$ independent coins and the occurrence of each word in the document is a Bernoulli event. Therefore, the generative portion of equation (2.16) is

$$P(d|c,\boldsymbol{\theta}) = \prod_{j=1}^{|V|} x_j \, P(\omega_j|c) + (1 - x_j)[1 - P(\omega_j|c)] \tag{2.17}$$

where $x_j = 1$ [0] means that word ω_j does [does not] occur in d and $P(\omega_j \mid c)$ is the probability of observing word ω_j in documents of class c. Here $\boldsymbol{\theta}$ represents the set of probabilities (or parameters) $P(\omega_j \mid c)$, which is the probability of the binary event that word ω_j is within class c. Alternatively, a document can be viewed as a sequence of events $W_1, \ldots, W_{|d|}$. Each observed W_t has a vocabulary entry (from 1 to $|V|$) as an admissible realization. In addition, since the document is a sequence, serial order among words should also be taken into account when modeling $P(W_1, \ldots, W_{|d|} \mid c)$. This could be done, for example, by using a Markov chain. A simplifying assumption, however, is that word occurrences are independent of their (relative) positions, given the class. Equivalently, it is assumed that the bag of words representation retains all the relevant information for assessing the probability of a document whose class is known. Under the word based event model, generating a document is like throwing a die with $|V|$ faces $|d|$ times, and the occurrence of each word in the document is a multinomial event. Hence, the generative portion of the model is a multinomial distribution.

$$P(d|\boldsymbol{\theta}) = GP(|d|)\prod_{j=1}^{|V|} P\left(\omega_j|c\right)^{n_j} \tag{2.18}$$

where n_j is the number of occurrences of word ωj in d, and $P(\omega_j \mid c)$ is the probability that word ω_j occurs at any position $t \in [1, \ldots, |d|]$; because of the bag of words assumption this does not depend on t . Here the parameters $\boldsymbol{\theta}$ are the set of probabilities $P(\omega_j \mid c)$, where now $\boldsymbol{\Sigma}_{j=1} |V| P(\omega_j \mid c) = 1$. McCallum and Nigam (1998) [31] found empirically that the multinomial model outperforms the Bernoulli model in several evaluation benchmarks. The bag of words assumption results in a factorization of $P(W_1, .$

. . , $W_{|d|} \mid c)$ explaining why the Naive Bayes is often used in the literature to refer to both event models. The normalization factor is the multinomial coefficient

$$G = \frac{|d|!}{\prod_j n_j!} \tag{2.19}$$

Neither $P(|d|)$ nor G are needed for classification, hence $|d|$, the number of words or terms in a document is assumed to be independent of the class. This last assumption can be removed and $P(|d| \mid c)$ explicitly modeled. Models of document length (e.g. based on Poisson distributions) have been used for example in the context of probabilistic retrieval. Note that in the case of the Bernoulli model there are $2|V|$ possible different documents, while in the case of the multinomial model there is an infinite (but countable) number of different documents.

An additional model that may be developed that lies somewhere in between the Bernoulli and the multinomial models consists of keeping the document based event model but extending Bernoulli distributions to integer distributions, such as the Poisson (Lewis 1998) [32]. Finally, extensions of the basic Naive Bayes approach that allow limited dependencies among features have also been proposed (Friedman and Goldszmidt 1996; Pazzani 1996) [33, 34]. However, these models are characterized by a larger set of parameters and may overfit the data (Koller and Sahami 1997) [35]. Teaching a Naive Bayes classifier consists of estimating the parameters $\boldsymbol{\theta}$ from the available data. It assumes that the training data set consists of a collection of labeled documents $\{(d_i, c_i), i = 1, \ldots, n\}$. In the Bernoulli model, the parameters $\boldsymbol{\theta}$ include $\theta c,j = P(\omega_j \mid c), j = 1, \ldots, |V|, c = 1, \ldots, K$. These are estimated as normalized sufficient statistics

$$\hat{\theta}_{c,j} = \frac{1}{N_c} \sum_{i:c_i:=c}^{n} x_{ij} \tag{2.20}$$

where $N_c = |\{i : c_i = c\}|$ and $x_{ij} = 1$ if ω_j occurs in d_i . Additional parameters are the class prior probabilities $\theta_c = P(c)$, which are estimated as

$$\hat{\theta}_c = \frac{N_c}{n} \tag{2.21}$$

The estimates above correspond to machine learning estimates of the parameters. Bayesian estimates can be quite useful when estimating parameters from sparse data. A model based on Machine Learning parameter estimates would assign a probability of zero to that document, irrespective of how well the other words in the document matched class *c*. On the other hand, a model based on Bayesian estimates would assign that word-class combination a low nonzero probability and still allow the other words to play a role in determining the final class prediction for the document. In the case of the multinomial model of equation (2.18), the generative parameters are $\theta_{c,j} = P(\omega_j \mid c)$. These parameters must satisfy $\Sigma_j \ \theta_{c,j} = 1$ for each class *c*. To estimate these parameters it is common practice to introduce Dirichlet priors. The resulting estimation equations are derived as follows. In the case of the distributions of terms given the class, a Dirichlet prior with hyperparameters q_j and α, results in the estimation formula

$$\hat{\theta}_{c,j} = \frac{\alpha q_j + \sum_{i:c_i=c}^{n} n_{ij}}{\alpha + \sum_{l=1}^{|V|} \sum_{i:c_i=c} n_{ij}} \tag{2.22}$$

where n_{ij} is the number of occurrences of ω_j in d_i . A simple non-informative prior assigns $q_j = 1/|V|$ and $\alpha = |V|$. Intuitively, this prior corresponds to the assumption that

each word is observed exactly once in one document of each class. This method (also known as Laplace smoothing) prevents the problem of estimating a null value for a parameter if a certain term ω_j never occurs in documents of a given class c in the training. Similarly, the estimation formula for the (unconditional) class probabilities is

$$\hat{\theta}_c = \frac{q'_c \alpha' + N_c}{\alpha' + n} \tag{2.23}$$

- Support Vector machines (SVMs)

Support vector machines (SVMs) were introduced in Cortes and Vapnik (1995) [35] to extend earlier seminal work by Vapnik on statistical learning theory. The basic underlying idea, often referred to as *structural risk minimization* is closely related to the theory of regularization but also to Bayesian approaches to learning (Evgeniou *et al.* 2000) [36] and is essentially guided by the principle that the hypothesis that explains a finite set of examples should be searched in an appropriately small hypothesis space.

SVMs are particularly well suited to deal with high dimensional data such as vector space representations of text documents. In their standard formulation, they deal with the binary classification problems where the number of classes is restricted to two.

Consider a training set $D = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, n\}$ with $\boldsymbol{x}_i \in \mathrm{R}^m$, and where $y_i \in \{-1, 1\}$ is an integer that specifies whether $\boldsymbol{x}_i$ is a positive or a negative example. A linear discriminant classifier is then defined by introducing the *separating hyperplane*

$$\{x : f(x) = w^T x + w_0 = 0\} \tag{2.24}$$

where $\boldsymbol{w} \in \mathrm{R}^m$ and $w_0 \in \mathrm{R}$ are adjustable coefficients that play the role of model parameters. A binary classification function $h : \mathrm{R}^m \rightarrow \{0, 1\}$ can be obtained by taking the sign of $f(\boldsymbol{x})$, i.e.

$$h(x) = \begin{cases} 1, & if\ f(x) > 0 \\ 0, & otherwise. \end{cases} \tag{2.25}$$

Learning in this class of models consists of determining w and w_0 from the data. The training examples are said to be *linearly separable* if there exists a hyperplane whose associated classification function is consistent with all the labels, i.e. if $y_i\ f(\boldsymbol{x}_i) > 0$ for each $i = 1, \ldots, n$. see Figure 2.9.

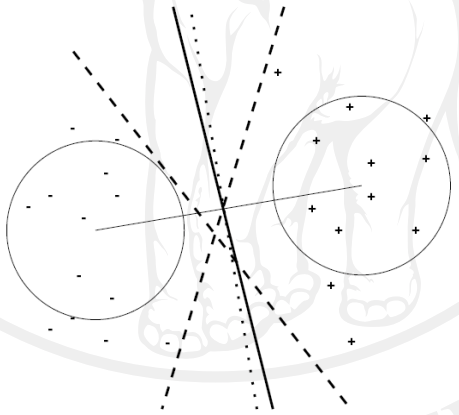

Figure 2.9 Alternative linear decision boundaries for a binary classification problem

Here, suppose that positive and negative examples are generated by two Gaussian distributions with the same covariance matrix and that positive and negative points are generated with the same probability. In such a setting, the optimal (Bayes) decision boundary is the one that minimizes the posterior probability that a new point is misclassified and, as it turns out, this boundary is the hyperplane that is orthogonal to the segment connecting the centers of mass of the two distributions (dotted line).

A random hyperplane that just happens to separate training points (dashed line) can be substantially far away from the optimal separation boundary, leading to poor generalization to new data. The difficulty grows with the dimensionality of the input space *m* since for a fixed *n* the set of separating hyperplane grows exponentially with *m* (a problem known as the *curse of dimensionality*). Remember that in the case of text categorization *m* may be significantly large (several thousands).

The statistical learning theory developed by Vapnik (1998) [37] shows that an *optimal separating hyperplane* (relative to the training set) has two important properties: it is unique for each linearly separable data set, and its associated risk of overfitting is smaller than for any other separating hyperplane. The *margin M* of the classifier is defined to be the distance between the separating hyperplane and the closest training examples. The optimal separating hyperplane is then the one having the maximum margin.

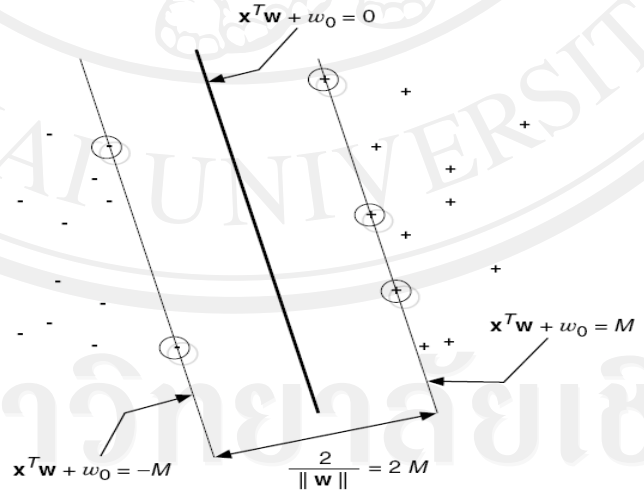

Figure 2.10 Illustration of the optimal separating hyperplane and margin. Circled points are support vectors.

From **Figure 2.10**, the theory suggests that the risk of overfitting for the maximum margin hyperplane (solid line) is smaller than for the dashed hyperplane. Indeed, in this example the maximum margin hyperplane is significantly closer to the Bayes optimal decision boundary. In order to compute the maximum margin hyperplane, the distance of a point $\boldsymbol{x}$ from the separating hyperplane is observed first,

$$\frac{1}{\|w\|}(w^T x + w_0) \tag{2.26}$$

Thus, the optimal hyperplane can be obtained by solving the constrained optimization problem.

$$\min_{w,w_0} M \; subject \; to \; \frac{1}{\|w\|} y_i (w^T x + w_0) \geq M, i = 1, \dots, n \tag{2.27}$$

The above problem can be transformed to its dual by first introducing the vector of Lagrangian multipliers $\boldsymbol{\alpha} \in \mathrm{R}^n$ and writing the Lagrangian function

$$\mathcal{L}(D) = -\frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} \alpha_i [y_i (w^T x + w_0) - 1] \tag{2.28}$$

and subsequently setting to zero the derivatives of (2.30) with respect to $\boldsymbol{w}$, w_0, obtaining

$$\max_{\alpha} -\frac{1}{2}\alpha^T \Lambda \alpha + \sum_{i=1}^{n} \alpha_i \; subject \; to \; \alpha_i \geq 0, i = 1, \dots, n \tag{2.29}$$

where $\boldsymbol{\Lambda}$ is an $n \times n$ matrix with $\lambda_{ij} = y_i y_j \boldsymbol{x_i^T} \boldsymbol{x_j}$. This is a quadratic programming (QP) problem that can be solved, in principle, using standard optimization packages. For each training example i,

$$\alpha_i[y_i(w^T x + w_0) - 1] = 0 \tag{2.30}$$

and therefore either $\alpha_i = 0$ or $y_i\ (\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0) = 1$. In other words, if $\alpha_i > 0$ then the distance of point $\boldsymbol{x}_i$ from the separating hyperplane must be exactly M (see Figure 2.8). Points with associated $\alpha_i > 0$ are called *support vectors*. The decision function $h(\boldsymbol{x})$ can be computed via equation (2.25) or, equivalently, from the following dual form:

$$f(x) = \sum_{i=1}^{n} y_i \alpha_i \; x^T x_i \tag{2.31}$$

It is important to point out that in the case of text classification it may be very important to exploit the sparseness of the data vectors while computing dot products in Equations (2.31) and (2.32). If the training data are not linearly separable, then this analysis can be generalized by introducing m nonnegative *slack variables* ξ_i and replacing the optimization problem in equation (2.29) with

$$\min_{w,w_0} \|w\| + c \sum_{i=1}^{n} \xi_i \text{ subject to } \begin{cases} y_i(w^T x_i + w_0) \geq 1 - \xi_i, & i = 1,,,,,n \\ \xi_i \geq 0, & i = 1,,,,,n \end{cases} \tag{2.32}$$

where the constant C controls the cost associated with misclassifications. This problem can also be dualized in the form.

$$\max_{\alpha} -\frac{1}{2}\alpha^T \Lambda \alpha + \textstyle\sum_{i=1}^{n} \alpha_i \; subject\ to\ 0 \leq \alpha_i \leq C,\ i = 1, \ldots, n \tag{2.33}$$

The classifier obtained in this way is commonly referred to as a support vector machine (SVMs).

If the data are considerably nonlinearly separable then an SVMs classifier will have a low accuracy, i.e. even the best linear hyperplane may be quite inferior in terms of prediction accuracy relative to a good nonlinear decision boundary. The methods developed in this section can be further extended to accommodate nonlinear separation by using kernel functions that map points $\boldsymbol{x} \in R^m$ into a higher dimensional space called the *feature space*, where data are linearly separable. Details on kernel methods can be found in Schoelkopf and Smola (2002) [38] and the extraction of conditional probabilities from multiclass SVMs is studied in Passerini *et al.* (2002) [39].

2) Feature selection for classification models generating

In the classification task, the appropriate feature selection leads to an effective classification model. In this work, Information Gain Measure, Mutual information measure and Odds ratio measures are selected to be the feature for the generating models.

- **Information gain measure**

Information gain (IG) measures the amount of information in bits in the class prediction, if the only information available is the presence of a feature and the corresponding class distribution. Concretely, it measures the expected reduction in entropy (uncertainty associated with a random feature) [40]. Given S_X the set of training examples, $\mathbf{x}_i$ the vector of the i^{th} variables in this set, $\left|S_{x_i=v}\right|/|S_x|$ is the fraction of examples of the i^{th} variable having the value

$$IG(S_X, X_i) = H(S_X) = \sum_{\boldsymbol{v=valules}(X_i)} \frac{\left|S_{x_i=v}\right|}{|S_x|} H(S_{x_i=v}) \text{ with entrophy} \quad (2.34)$$

where $H(S_X)$ = $p\pm(S)$ is the probability of a training example in the set S to be of the positive/negative class.

- **Mutual information measure**

Mutual information, which is also known as information gain or best individual feature, perhaps is the most naïve measurement. For the candidate feature f, its criterion function of MI is

$$J(f)=I(C; f)=H(f)-H(f|C) \quad (2.35)$$

where $H(f)$ is information entropy of f and $H(f|C)$ is its conditional value with respect to C. That is to say, α=1, $g(C, f, S)=I(C; f)$ and δ = 0. This method chooses the best individual features out of the feature selection procedure [41]. It first evaluates all candidate features individually according to the criterion function $J(f)$, and then sorts them in descending order in terms of $J(f)$. After that, the best k features are picked out to take the place of the whole feature set.

- **Odds ratio measure**

The Odds Ratio reflects the odds of a word occurring in the positive class normalized by that of the negative class. It has been used for relevance ranking in information retrieval.

Let $P(t|c)$ be the probability of a randomly chosen word being t given that the document it was chosen from belongs to a class c. Then $odds(t|c)$ is defined as $P(t|c)/[1-P(t|c)]$ and the odds ratio is equal to

$$OR(t) = \ln[odds(t|c_+)/odds(t|c_-)] \quad (2.36)$$

This scoring measure favors features that are representative of positive examples. As a result a feature that occurs very few times in positive documents but never occurs in negative documents will get a relatively high score. Thus, many features that are rare among the positive documents will be ranked at the top of the feature list. Odds ratio is known to work well in combination with Naïve Bayes [42]

3) Dataset

480 from the NCBI corpus were used as the dataset. They were part of the group of papers related to (Bio Medical Text) bioinformatics. This dataset is composed of 4 classes, Evolution, Genetics, Genomes_studies and Molecular Biology.

2.4.3 Results and Discussion

The classification models were implemented using the R programming language. The results of the classification model predictions are shown in **Table 2.2**

Table 2.2 The performance of classifications models with features selections

	Information Gain (IG) Measure (%)	**Mutual Information (MI) Measure (%)**	**Odds Ratio Measure (%)**
***k*-Nearest neighbors**	25	24	25
Naive Bayes	50	49	49
Support Vector Machines (SVMs)	**75**	**73**	**75**

From **Table 2.2**, the classification model generated by the SVMs returned the highest score prediction while the *k*-nearest neighbor classification model returned the

lowest score prediction. The feature selection measure, IG measure and Odds Ratio Measure returned higher score prediction than the MI Measure.

The objective of this work was the preliminary review and comparison of bioinformatics-text classification models which were generated by using machine learning techniques; *k*-nearest neighbor, Naive Bayes and SVMs. The results showed that the SVMs model was suitable for generating the classification model and feature selection still remained an important process for text mining.

2.5 Conclusion

In this chapter, general terms from the biological literature were studied via machine learning techniques. The results show that the biological texts characteristics are different from others. Even though, collaborate with the LSI which is a statistical technique that attempts to estimate the hidden structure that generates terms given concepts to discovers the most important associative patterns between words and concepts. The results still show that general terms could represent each class well. In the classification task, three classic feature selections techniques were selected to study. We found that feature selection methods (on Odds Ratio IG and MI) based on computational approaches were not enough for biological text in information extraction while the classification model performance, SVMs is particularly well suited to deal with high dimensional data such as vector space representations of text documents than the others. As these reason, features are important for to be the factor to generate the prediction

model. Thus the next chapter, the suitable feature selection from biological literature is considered to improve the information extraction.