Automated Discovery of Abstracts Reporting Gene-Disease Relationship

By
Divya Sharma
MT18235

Under the Supervision of Dr. Debarka Sengupta

Indraprastha Institute of Information Technology Delhi

January, 2020

Automated Discovery of Abstracts Reporting Gene-Disease Relationship

By
Divya Sharma
MT18235

Under the Supervision of **Dr. Debarka Sengupta**

Submitted

In partial fulfilment of the requirements for the degree of

Master of Technology

To

Indraprastha Institute of Information Technology Delhi

January, 2020

Certificate

This is to certify that the thesis titled **"Automated Discovery of Abstracts Reporting Gene-Disease Relationship"** is being submitted by **Divya Sharma** to the Indraprastha Institute of Information Technology Delhi, for the award of the **Master of Technology**, is an original work carried out by her under my supervision. In my opinion, the project has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this project have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

January, 2020

Dr. Debarka Sengupta

Department of Computation Biology

Indraprastha Institute of Information Technology Delhi

New Delhi 110 020

Acknowledgments

I would like to express my warm gratitude to my supervisor, Dr. Debarka Sengupta for exposing me to this wonderful topic of research and guiding me throughout. I am grateful to him for our interesting discussions about the problem I studied and also for his active participation. I would also like to thank the Centre for Computational Biology, IIIT Delhi and IT administrators at IIIT Delhi for providing me with all the necessary resources. I especially thank Dr. Tanmoy Chakraborty, who in a short stint provided extremely valuable inputs and provided critical feedback.

I would also like to thank my family and friends for providing much-needed support. Their faith and confidence in me and their constant encouragement has helped me tremendously.

Abstract

Text classification is a construction problem of models which can classify new documents into predefined classes. It is important before text mining that we know what is the most important data that we require for our research. Text mining has become an essential tool for biomedical research. Our project aims to identify the gene-disease relationship using natural language processing techniques and word embeddings. Assignment of high-dimensional vectors (embeddings) to words in a text corpus in a way that preserves their syntactic and semantic relationships is one of the most fundamental techniques in natural language processing (NLP). We present a completely generic model based on statistical word embeddings, which shows the gene similarity and proves the gene-disease relationship using word analogies.

Table of contents

List of Figures

List of Tables

Chapter 1

1 Introduction

Classification of pubmed abstracts was done for getting relevant abstracts related to gene-disease relationships. Manual and automatic text classification was used to classify abstracts. In manual classification, the content of the text was interpreted and categorized accordingly to get the training data. In automatic text classification machine learning based systems were used. By using pre-labeled examples as training data, machine learning algorithm was applied that learned the different associations between pieces of text and that a particular output (i.e. tags) was expected for a particular input (i.e. text). The first step towards training a classifier with machine learning is feature extraction: a method was used to transform each text into a numerical representation in the form of a vector. Then, the machine learning algorithm was fed with training data that consists of pairs of feature sets (vectors for each text example, abstracts annotated manually) and tags (e.g. relevant, irrelevant) to produce a classification model.

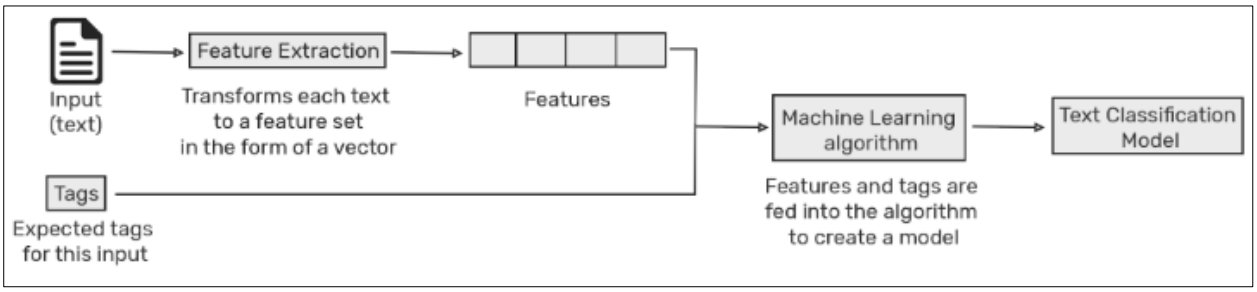

Figure 1.1 Building the text classification model

After training with enough training samples, the machine learning model began to make accurate predictions. The same feature extractor was used to transform unseen text to feature sets which were fed into the classification model to get predictions on tags.

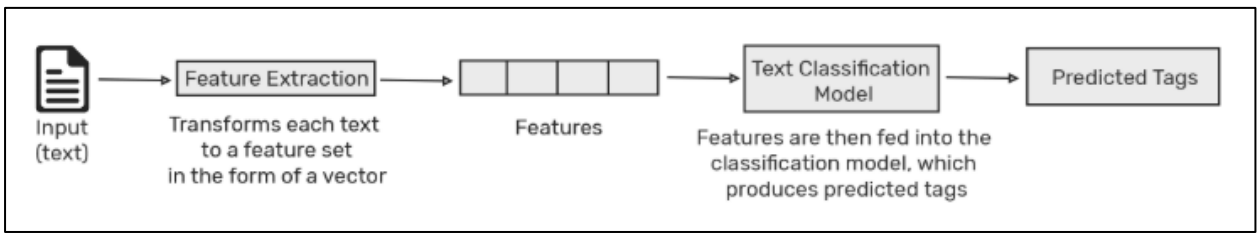

Figure 1.2 Predicting tags from the model

Pubmed abstracts were annotated manually on the basis of gene-disease relationships to get the training data. For the annotated set various classifiers were experimented like, svc, logistic regression, and decision tree. Svc was finally used to filter out papers from the entire set (about

1mn out of 3mn papers were found to be relevant) as it showed better accuracy. All the data was cleaned, preprocessed, and converted into vectors. After classifying the relevant and irrelevant abstracts word embeddings were produced by applying word2vec. Some useful biological results were obtained after the application of word embeddings.

Chapter 2

2 Background and Related Work

Text classification is a construction problem of models which can classify new documents into predefined classes [1]. Currently, it is a sophisticated process involving not only the training of models, but also numerous additional procedures, e.g. data pre-processing, transformation, and dimensionality reduction. Text classification remains a prominent research topic, utilizing various techniques and their combinations in complex systems. Furthermore, researchers are either developing new classification systems or improving the existing ones, including their elements to yield better results, i.e. a higher computational efficiency [1].

A massive amount of hidden information is present in the biological abstracts which need to be retrieved to gather new insights into the era. Most of the knowledge is buried in the form of unstructured textual information only available as written articles. A large volume of data is available as abstracts that require to be analyzed collectively. For achieving this, we encoded the biologically published literature as information-dense word embeddings. These embeddings capture gene-disease relationship concepts without any explicit insertion of biological knowledge such as the genes associated with a particular disease, symptoms associated with a disease, etc. Our findings showed the possibility of extracting relationships among genes and diseases from the massive body of scientific abstracts in a collective manner.

In recent years, several word embedding models and pre-trained word embeddings [2], [3], [4] have been made publicly available and successfully applied to many biomedical NLP (BioNLP) tasks. But this technique is not applied for finding out gene-disease relationships yet. Moreover, context-specific interaction networks from vector representation of words have been identified for prostate cancer, it has not been applied to a wider area [5]. Either the amount of data taken is very less [6] or the context of the application is very limited. BioWordVec: an open set of biomedical word vectors/embeddings that combines subword information from unlabeled biomedical text with a widely-used biomedical controlled vocabulary called Medical Subject Headings (MeSH) used 27,599,238 pubmed articles to create the word embeddings and applying deep learning and classification models on it [7].

Unsupervised word embeddings captured latent knowledge from materials science literature. This paper showed how applying word embedding techniques led to the discovery of a novel property of a metal that was not considered as thermoelectric earlier but possesses high cosine similarity with the word thermoelectric [8]. They removed abstracts that were not relevant to their area by training a binary classifier that could label abstracts as ‘relevant’ or ‘not relevant’. We tried to follow a somewhat similar procedure to classify the data.

Chapter 3

3 Methodology

3.1 Data Collection

We collected approximately 3 million abstracts primarily focussed on molecular biology literature and genetics. A list of keywords (approximately 500) that were relevant to our area of research was provided to them. All the articles were from NCBI (The National Center for Biotechnology Information) pubmed. Abstracts were taken from the site [9]. We first tried to get the data through web-scraping using beautifulsoup python library but could scrape only around 100,000 articles. All the data that we got was in the form of '.json' files.

3.1.1 Reading json files:

All of the json data is converted into a readable format by using python package json.
Introduction of JSON in Python :
The full-form of JSON is JavaScript Object Notation. It means that a script (executable) file which is made of text in a programming language, is used to store and transfer the data. Python supports JSON through a built-in package called json. To use this feature, we import the json package in Python script. The text in JSON is done through quoted-string which contains the value in key-value mapping within { }. It is similar to the dictionary in Python. JSON shows an API similar to users of Standard Library marshal and pickle modules and Python natively supports JSON features. JSON supports primitive types, like strings and numbers, as well as nested lists, tuples and objects.
Serializing JSON :
The process of encoding JSON is usually called serialization. This term refers to the transformation of data into a series of bytes (hence serial) to be stored or transmitted across a network. To handle the data flow in a file, the JSON library in Python uses dump() function to convert the Python objects into their respective JSON object, so it makes easy to write data to files. See the following table given below.

PYTHON OBJECT	JSON OBJECT
dict	object
list, tuple	array
str	string
int, long, float	numbers
True	true
False	false
None	null

Table 3.1 Python object's corresponding json object

3.1.1.1 Deserializing JSON :

Deserialization is the opposite of Serialization, i.e. conversion of JSON object into their respective Python objects. The load() method is used for it. If we have used Json data from another program or obtained as a string format of Json, then it can easily be deserialized with load(), which is usually used to load from the string, otherwise, the root object is in list or dict.

Different functions for json:

json.load(fp, *, cls=None, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, object_pairs_hook=None, **kw)

Deserialize fp (a .read()-supporting text file or binary file containing a JSON document) to a Python object using this conversion table.

object_hook is an optional function that will be called with the result of any object literal decoded (a dict). The return value of object_hook will be used instead of the dict. This feature can be used to implement custom decoders (e.g. JSON-RPC class hinting).

object_pairs_hook is an optional function that will be called with the result of any object literal decoded with an ordered list of pairs. The return value of object_pairs_hook will be used instead of the dict. This feature can be used to implement custom decoders. If object_hook is also defined, the object_pairs_hook takes priority.

parse_float, if specified, will be called with the string of every JSON float to be decoded. By default, this is equivalent to float(num_str). This can be used to use another datatype or parser for JSON floats (e.g. decimal.Decimal).

parse_int, if specified, will be called with the string of every JSON int to be decoded. By default, this is equivalent to int(num_str). This can be used to use another datatype or parser for JSON integers (e.g. float).

parse_constant, if specified, will be called with one of the following strings: '-Infinity', 'Infinity', 'NaN'. This can be used to raise an exception if invalid JSON numbers are encountered.

Changed in version 3.1: parse_constant doesn't get called on 'null', 'true', 'false' anymore.

To use a custom JSONDecoder subclass, specify it with the cls kwarg; otherwise, JSONDecoder is used. Additional keyword arguments will be passed to the constructor of the class.

If the data being deserialized is not a valid JSON document, a JSONDecodeError will be raised.

Changed in version 3.6: All optional parameters are now keyword-only.

Changed in version 3.6: fp can now be a binary file. The input encoding should be UTF-8, UTF-16 or UTF-32.

3.2 Extracting relevant columns from the data

Our data was extracted from NCBI (The National Center for Biotechnology Information) pubmed. It contained various columns such as Sr.No., pubmed id (PMID), abstract, title, doi id, etc. PubMed comprises over 30 million citations for biomedical literature from MEDLINE, life science journals, and online books. PubMed citations and abstracts include the fields of biomedicine and health, covering portions of the life sciences, behavioral sciences, chemical sciences, and bioengineering. PubMed also provides access to additional relevant web sites and links to the other NCBI molecular biology resources. PubMed is a free resource that is developed and maintained by the National Center for Biotechnology Information (NCBI), at the U.S. National Library of Medicine (NLM), located at the National Institutes of Health (NIH). Various fields included in the pubmed json data are as follows:
Abstract (AB), Copyright Information (CI), Affiliation (AD), Article Identifier (AID), Author (AU), Create Date (CRDT), Date Completed (DCOM), Date Created (DA), Date Last Revised (LR), Date of Publication (DP), Edition (EN), Entrez Date (EDAT), Journal Title (JT), MeSH Terms (MH), NLM Unique ID (JID), PubMed Central Identifier (PMC), PubMed Unique Identifier (PMID), Title (TI).
Although we were interested in only a few of the above-mentioned fields. These fields are PubMed Unique Identifier (PMID), Title (TI) and abstract (AB). So, only these columns were extracted from the json data.

3.3 Annotating the data

The data were annotated manually. The pubmed abstract's content was read manually and interpreted. The interpretation was based on our area of research, i.e., molecular biology and genetics. The abstracts were classified as positive and negative keeping in mind the problems related to gene-disease relationships. The positive abstracts contained the information about the genes and the corresponding diseases, all other abstracts were taken as negative. For positive data the articles were taken from GWAS (genome-wide association study), COSMIC (Catalogue Of Somatic Mutations In Cancer) and OMIM (Online Mendelian Inheritance in Man) databases.

3.3.1 GWAS

In genetics, a genome-wide association study (GWA study, or GWAS), also known as whole-genome association study (WGA study, or WGAS), is an observational study of a genome-wide set of genetic variants in different individuals to see if any variant is associated with a trait. GWASs typically focus on associations between single-nucleotide polymorphisms (SNPs) and traits like major human diseases, but can equally be applied to any other genetic variants and any other organisms.

3.3.2 COSMIC

COSMIC is an online database of somatically acquired mutations found in human cancer. Somatic mutations are those that occur in non-germline cells that are not inherited by children. COSMIC, an acronym of Catalogue Of Somatic Mutations In Cancer, curates data from papers in the scientific literature and large scale experimental screens from the Cancer Genome Project at the Sanger Institute. The database is freely available to academic researchers and commercially licensed to others.

3.3.3 OMIM

Online Mendelian Inheritance in Man (OMIM) is a continuously updated catalog of human genes and genetic disorders and traits, with a particular focus on the gene-phenotype

relationship. As of 28 June 2019, approximately 9,000 of them over 25,000 entries in OMIM represented phenotypes; the rest represented genes, many of which were related to known phenotypes.

For annotating the abstracts a google sheet was prepared to contain the columns as, PMID (pubmed id), Year of publication, Disease/Problem defined, Database to which the article belongs. PMID was used to extract the abstracts from the data. The year of publication was taken to check that the positive data is not biased with respect to the publication years. A total of 3000 abstracts were annotated out of which 1500 are positive and 1500 are negative. Positive abstracts are the abstracts that were relevant to our area of research and all other abstracts are negative abstracts.

3.3.4 Checking the bias

For checking the bias total number of count of publications in a particular year was compared with the total number of positive publications and the total number of negative publications respectively in that year. The years were starting from 1900 till the present. A correlation between the total count and positive data was obtained that was 0.84 and the correlation between the count and the negative data was 0.92. This showed that our data was not biased.

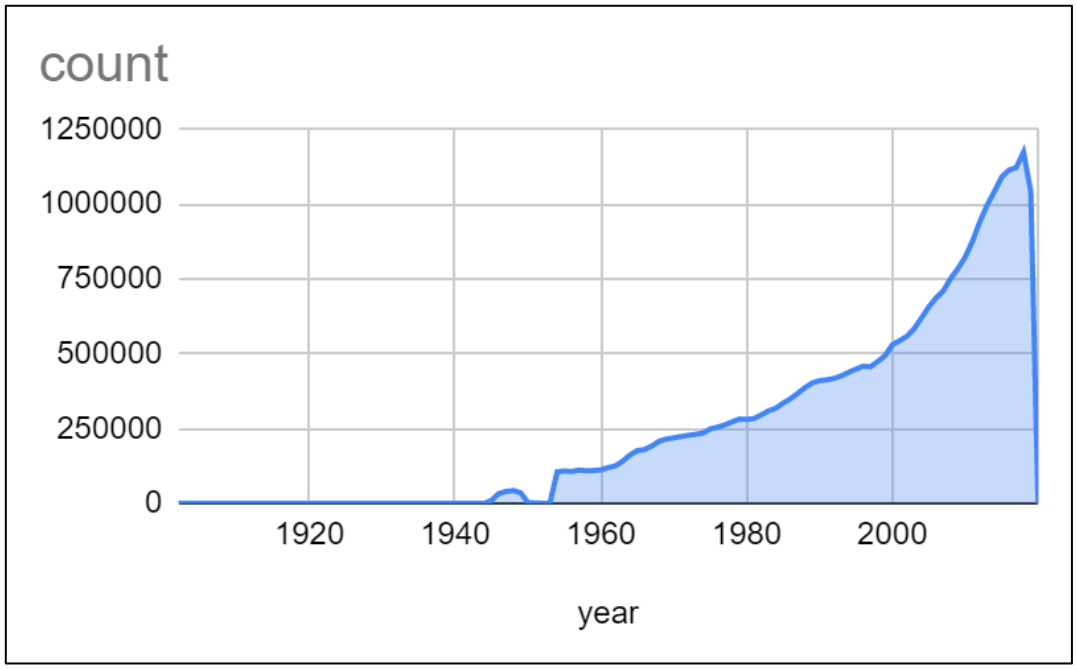

Figure 3.1 Total number of publications in respective years.

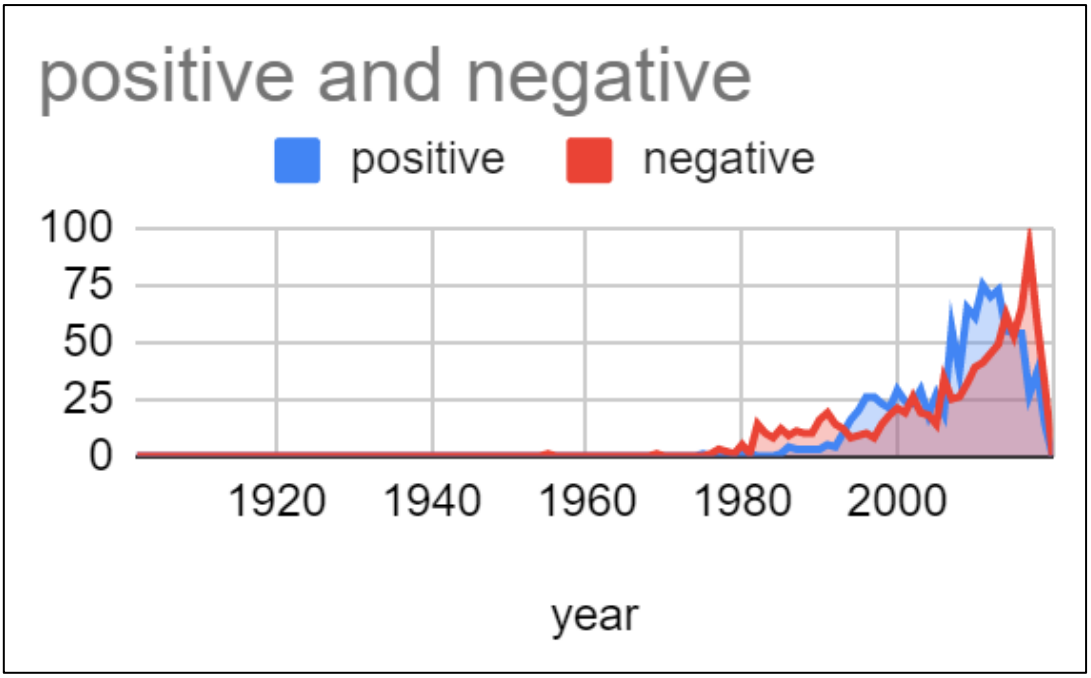

Figure 3.2 Number of positive abstracts and negative abstracts in respective years.

3.4 Cleaning and preprocessing the data

Only abstracts were taken as the data, all other fields such as title, PMID, Keywords, etc. were removed. To preprocess our text simply means to bring our text into a form that is predictable and analyzable for our task. A task here is a combination of approach and domain.
Steps involved in preprocessing

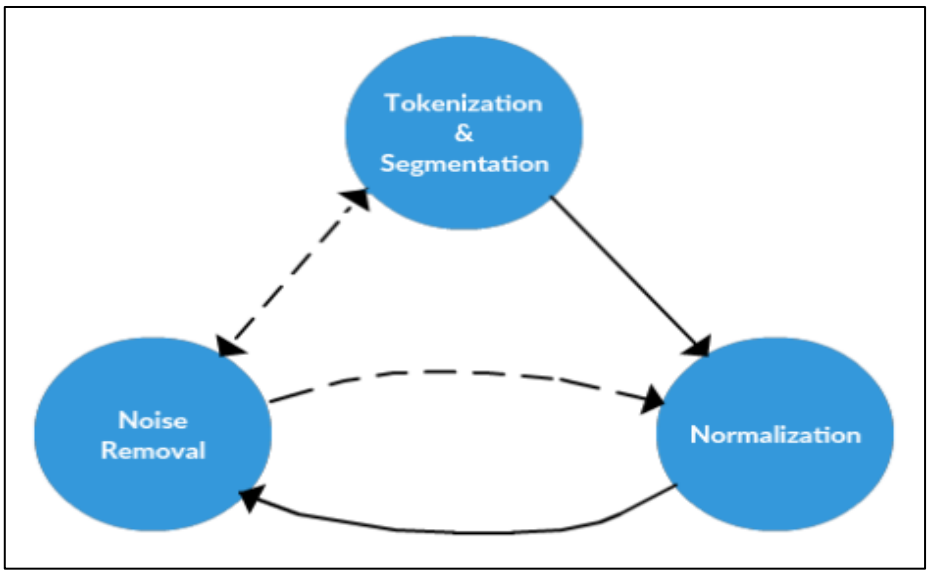

Figure 3.3 The text data preprocessing framework

3.4.1 Noise Removal.

Noise removal is about removing characters digits and pieces of text that can interfere with our text analysis. Noise removal is one of the most essential text preprocessing steps. It is also highly domain-dependent. For example, in Tweets, noise could be all special characters except hashtags as it signifies concepts that can characterize a Tweet. The problem with noise is that it can produce results that are inconsistent in our downstream tasks.

3.4.1.1 Removing HTML tags.

Often, the unstructured text contains a lot of noise, especially if you use techniques like web or screen scraping. HTML tags are typically one of these components which don't add much value towards understanding and analyzing text.

3.4.1.2 Replacing contractions.

While not mandatory to do at this stage prior to tokenization, replacing contractions with their expansions can be beneficial at this point since our word tokenizer will split words like "didn't" into "did" and "n't." It's not impossible to remedy this tokenization at a later stage, but doing so prior makes it easier and more straightforward.

3.4.2 Tokenization.

Tokenization is a step that splits longer strings of text into smaller pieces or tokens. Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc. Further processing is generally performed after a piece of text has been appropriately tokenized. Tokenization is also referred to as text segmentation or lexical analysis. Sometimes segmentation is used to refer to the breakdown of a large chunk of text into pieces larger than words (e.g. paragraphs or sentences), while tokenization is reserved for the breakdown process which results exclusively in words.

3.4.3 Normalization.

Text normalization is the process of transforming a text into a canonical (standard) form. For example, the word "gooood" and "gud" can be transformed into "good", its canonical form. Another example is mapping of near-identical words such as "stopwords", "stop-words" and "stop words" to just "stopwords".

3.4.3.1 Removing non-ASCII characters.

Non-ASCII characters are replaced with space.

3.4.3.2 Converting text corpus into lowercase.

All of the text is converted into lowercase so that every word could be read equally.

3.4.3.3 Removing punctuations.

All the punctuations were removed so that only words remain in the corpus.

3.4.3.4 Replacing numbers with words.

All the numbers were replaced with the corresponding English alphabet and converted them into words.

3.4.3.5 Removing STOP words.

Stopwords are the most common words in any natural language. For the purpose of analyzing text data and building NLP models, these stopwords might not add much value to the meaning of the document. Consider this text string – "There is a pen on the table". Now, the words "is", "a", "on", and "the" add no meaning to the statement while parsing it. Whereas words like "there", "book", and "table" are the keywords and tell us what the statement is all about.

3.4.4 Lemmatization.

Lemmatization reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called Lemma. A lemma (plural lemmas or lemmata) is the canonical form, dictionary form, or citation form of a set of words. For example, runs, running, ran are all forms of the word run, therefore run is the lemma of all these words.
Python NLTK library was used for the preprocessing. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning and wrappers for industrial-strength NLP libraries.

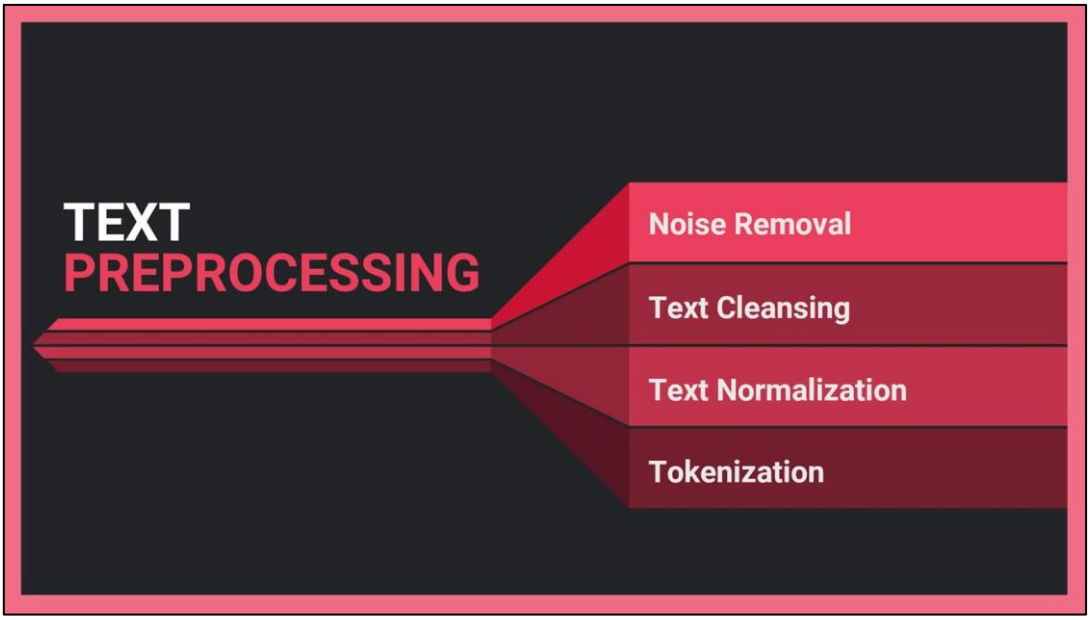

Figure 3.4 Text preprocessing steps

3.5 Converting the data into vectors

All the abstracts were converted to vectors using "BioSentVec_Pubmed_MIMICIII-bigram_d700.bin" file. It is freely available on NCBI PubMed. It contains the vectors of biomedical data. These are the pre-trained sentence embeddings for biomedical texts. These sentence embeddings are trained with over 30 million documents from both scholarly articles in PubMed and clinical notes in the MIMIC-III Clinical Database [10]. All the vectors for all the words were present in these embeddings.

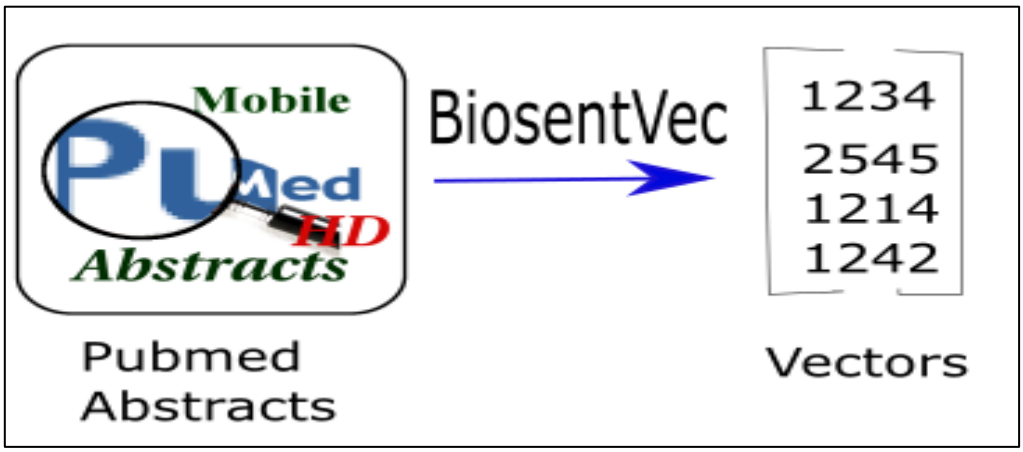

Figure 3.5 Text data is converted to vectors using BiosentVec

3.6 Supervised Classification: Applying different classifiers on the annotated data

Different classifiers were applied to the data to classify the abstracts as relevant and non-relevant. Relevant abstracts were the abstracts that were related to our area of research, that is, molecular biology and genetics. We aim to find out the relationship between genes and diseases through our model. All the abstracts other than the relevant abstracts were the non-relevant abstracts. Different classifiers that were applied are SVM (Support Vector Machines), Logistic Regression and Decision trees. Out of all of the above SVM classifiers best classified our abstracts. SVM showed the best accuracy out of all the three classifiers.

3.6.1 Support Vector Machines

Generally, Support Vector Machines is considered to be a classification approach but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.

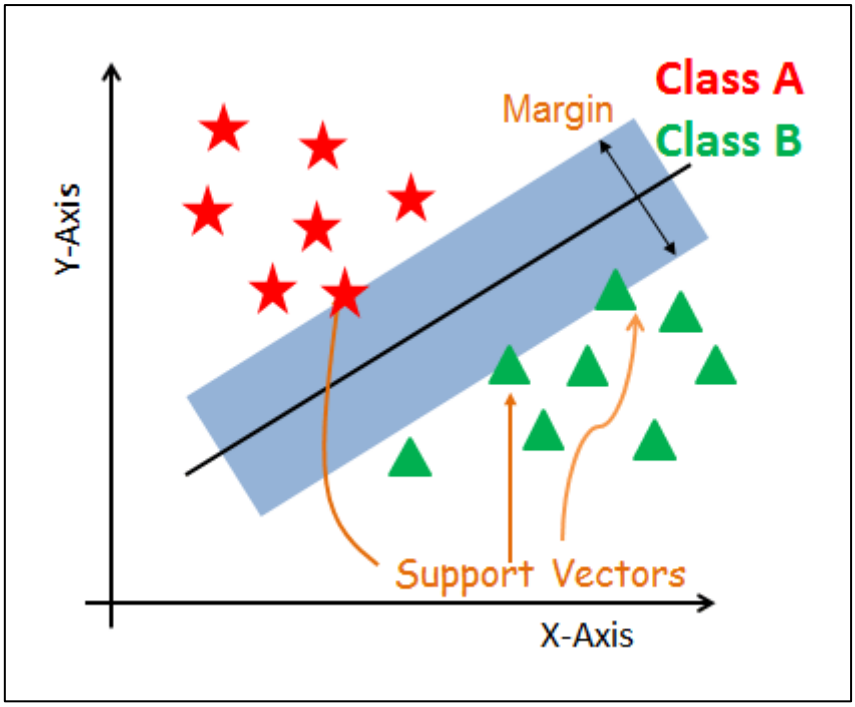

Figure 3.6 Finding a maximum marginal hyperplane

3.6.1.1 Support Vectors

Support vectors are the data points, which are closest to the hyperplane. These points will define the separating line better by calculating margins. These points are more relevant to the construction of the classifier.

3.6.1.2 Hyperplane

A hyperplane is a decision plane that separates between a set of objects having different class memberships.

3.6.1.3 Margin

A margin is a gap between the two lines on the closest class points. This is calculated as the perpendicular distance from the line to support vectors or closest points. If the margin is larger in between classes, then it is considered a good margin, a smaller margin is a bad margin.

3.6.1.4 How does SVM work?

The main objective is to segregate the given dataset in the best possible way. The distance between the nearest points is known as the margin. The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal hyperplane in the following steps:

1. Generate hyperplanes that segregate the classes in the best way. Left-hand side figure showing three hyperplanes black, blue and orange. Here, the blue and orange have higher classification errors, but the black is separating the two classes correctly.
2. Select the right hyperplane with the maximum segregation from either nearest data points as shown in the right-hand side figure.

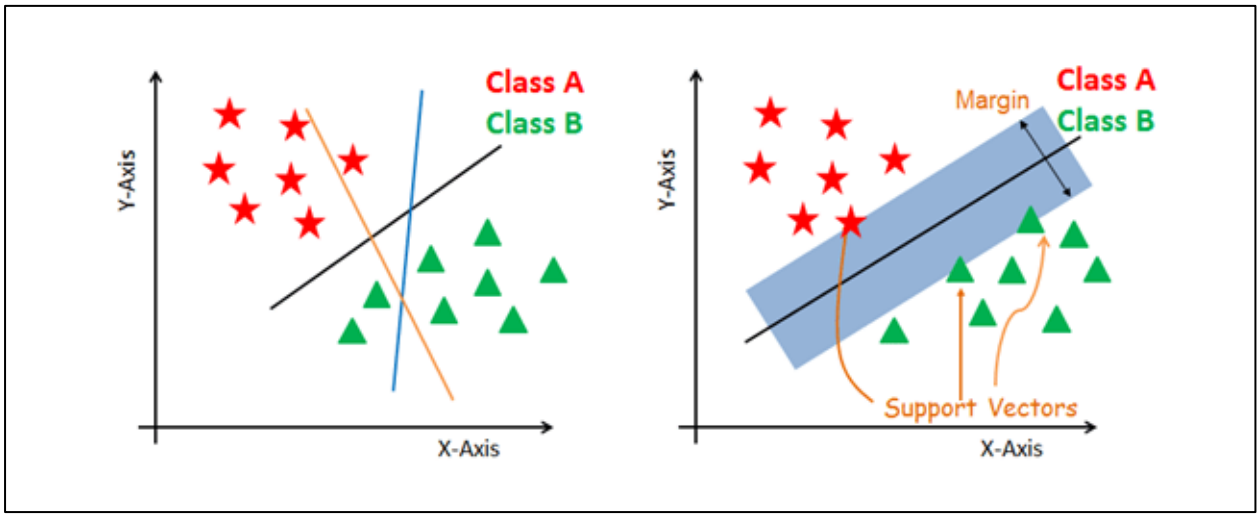

Figure 3.7 Drawing hyperplanes to segregate the classes

3.6.1.5 Dealing with non-linear and inseparable planes

Some problems can't be solved using linear hyperplane, as shown in the figure below (left-hand side).
In such a situation, SVM uses a kernel trick to transform the input space to a higher dimensional space as shown on the right. The data points are plotted on the x-axis and z-axis (Z is the squared sum of both x and y: z=x^2=y^2). Now you can easily segregate these points using linear separation.

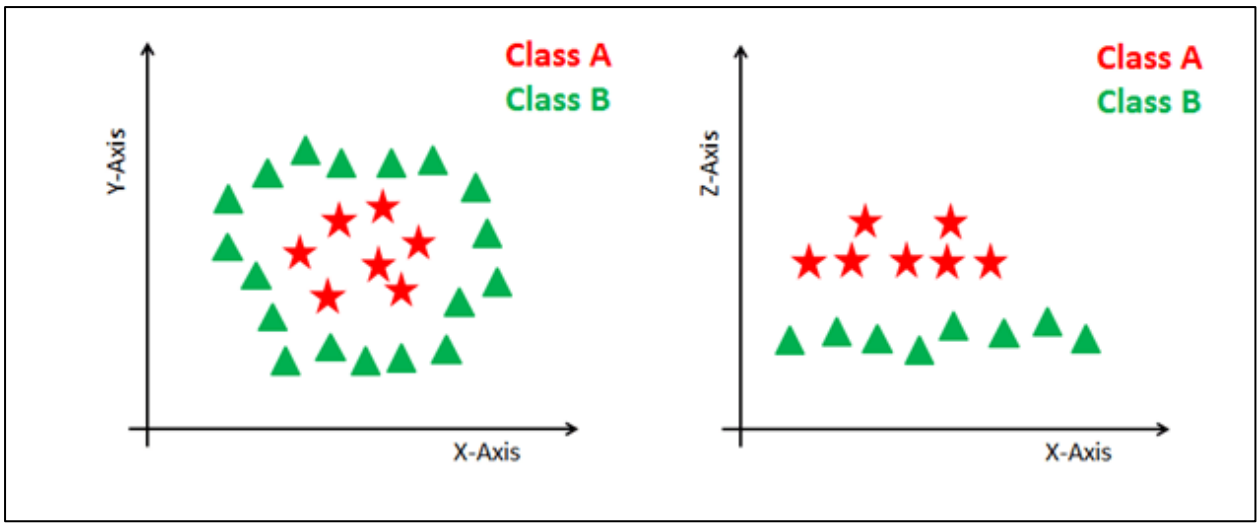

Figure 3.8 Non-linear and inseparable planes

3.6.1.6 SVM Kernels

The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a higher-dimensional space. In other words, you can say that it converts nonseparable problems to separable problems by adding

more dimension to it. It is most useful in a non-linear separation problem. Kernel trick helps you to build a more accurate classifier.

3.6.1.6.1 Linear Kernel

A linear kernel can be used as a normal dot product any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.
K(x, xi) = sum(x * xi)

3.6.1.6.2 Polynomial Kernel

A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.
K(x,xi) = 1 + sum(x * xi)^d
Where d is the degree of the polynomial. d=1 is similar to the linear transformation. The degree needs to be manually specified in the learning algorithm.

3.6.1.6.3 Radial Basis Function Kernel

The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite-dimensional space.
K(x,xi) = exp(-gamma * sum((x – xi^2))
Here gamma is a parameter, which ranges from 0 to 1. A higher value of gamma will perfectly fit the training dataset, which causes over-fitting. Gamma=0.1 is considered to be a good default value. The value of gamma needs to be manually specified in the learning algorithm.

3.6.1.7 Mathematical formulation

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite-dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

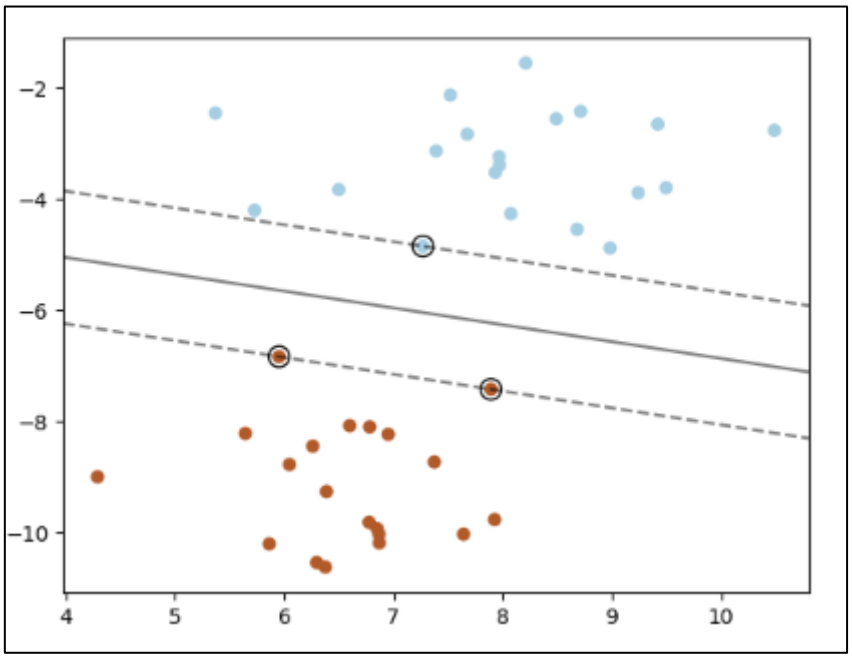

Figure 3.9 Mathematical formulation of hyperplanes

SVC

Given training vectors $x_i \in \mathbb{R}^p$, i=1,..., n, in two classes, and a vector $y \in \{1, -1\}^n$, SVC solves the following primal problem:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$
$$\text{subject to} y_i (w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, \ldots, n$$

Its dual is

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$
$$\text{subject to} y^T \alpha = 0$$
$$0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

Where e is the vector of all ones, C>0 is the upper bound, Q is an n by n positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. Here training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ.
The decision function is:

$$\text{sgn}(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho)$$

3.6.1.8 Advantages

SVM Classifiers offer good accuracy and perform faster prediction compared to the Naïve Bayes algorithm. They also use less memory because they use a subset of training points in the decision phase. SVM works well with a clear margin of separation and with high dimensional space.

3.6.1.9 Disadvantages

SVM is not suitable for large datasets because of its high training time and it also takes more time in training compared to Naïve Bayes. It works poorly with overlapping classes and is also sensitive to the type of kernel used.

3.6.1.10 Implementation

Python sklearn package was used.
We divided our data in the ratio 75:25 (training: testing).

The kernel used was ‘rbf’.
SVC(kernel = 'rbf', random_state = 0, gamma = "auto", probability = True)
These parameters were used while applying the classifier.

3.6.2 Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name. The unit of measurement for the log-odds scale is called a logit, from the logistic unit, hence the alternative names. Analogous models with a different sigmoid function instead of the logistic function can also be used, such as the probit model; the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each independent variable having its own parameter; for a binary dependent variable, this generalizes the odds ratio.

The binary logistic regression model has two levels of the dependent variable: categorical outputs with more than two values are modeled by multinomial logistic regression, and if multiple categories are ordered, by ordinal logistic regression, for example, the proportional odds ordinal logistic model. The model itself simply models probability of output in terms of input, and does not perform statistical classification (it is not a classifier), though it can be used to make a classifier, for instance by choosing a cutoff value and classifying inputs with probability greater than the cutoff as one class, below the cutoff as the other; this is a common way to make a binary classifier.

3.6.2.1 Definition of the logistic function:

An explanation of logistic regression can begin with an explanation of the standard logistic function. The logistic function is a sigmoid function, which takes any real input ‘t’, and outputs a value between zero and one; for the logit, this is interpreted as taking input log-odds and having output probability.
The standard logistic function is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

Let us assume that ‘t’ is a linear function of a single explanatory variable ‘x’. We can then express ‘t’ as follows:

$$t = \beta_0 + \beta_1 x$$

And the general logistic function can now be written as:

$$p(x) = \sigma(t) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

In the logistic model, p(x) is interpreted as the probability of the dependent variable Y equaling a success/case rather than a failure/non-case

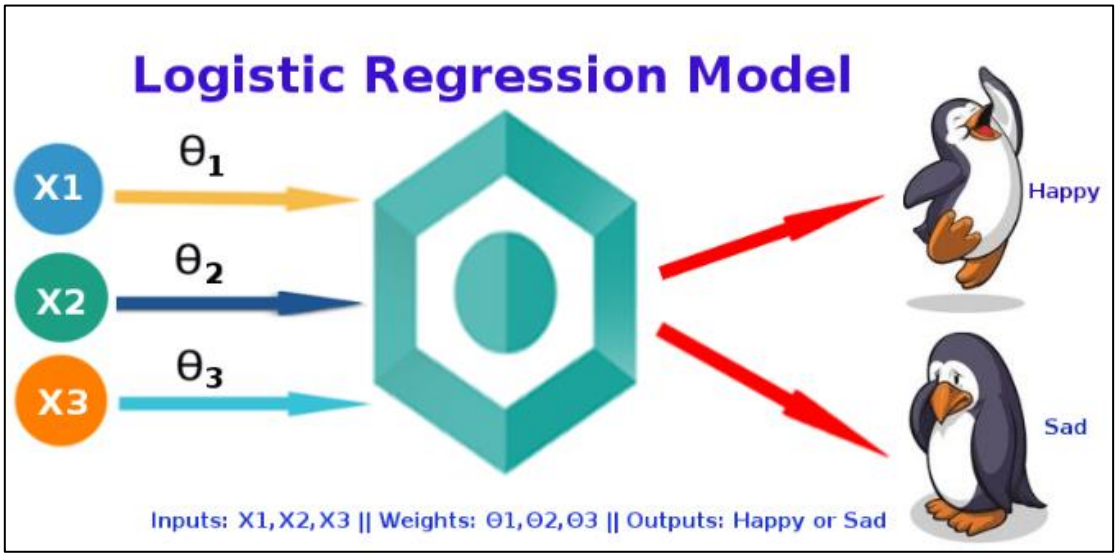

Figure 3.10 Representation of logistic regression model

3.6.2.2 Implementation:

Python sklearn, LogisticRegression library was used as the classifier.

3.6.3 Decision Trees

The decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like a tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

3.6.3.1 Construction of Decision Tree :

A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifiers does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

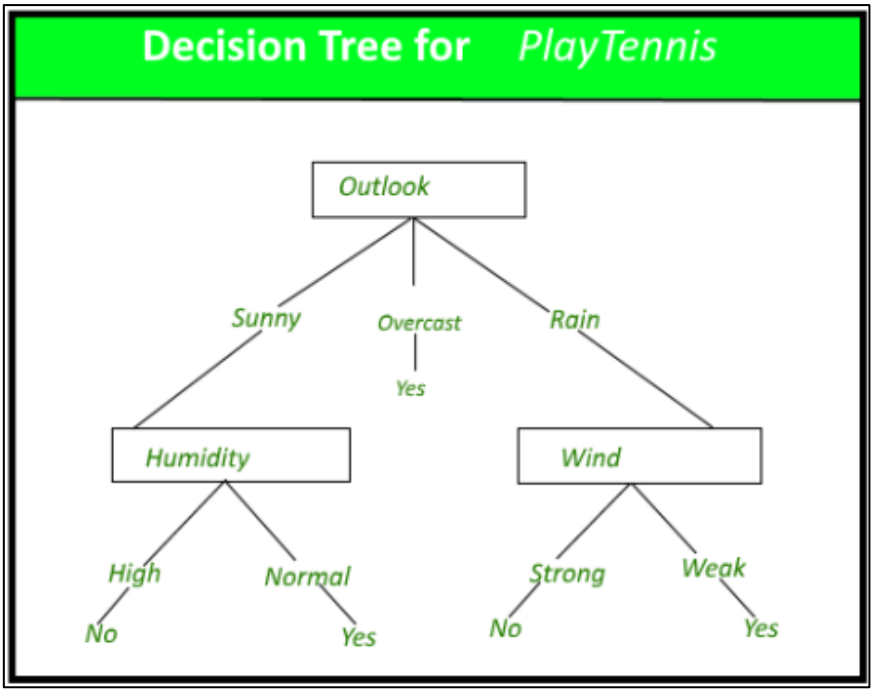

Figure 3.11 Decision tree for playing tennis

3.6.3.2 Decision Tree Representation :

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.
The decision tree in the above figure classifies a particular morning according to whether it is suitable for playing tennis and returning the classification associated with the particular leaf. (in this case Yes or No).
For example, the instance

(Outlook = Rain, Temperature = Hot, Humidity = High, Wind = Strong)

would be sorted down the leftmost branch of this decision tree and would, therefore, be classified as a negative instance.

In other words, we can say that the decision tree represents a disjunction of conjunctions of constraints on the attribute values of instances.

(Outlook = Sunny ^ Humidity = Normal) v (Outllok = Overcast) v (Outlook = Rain ^ Wind = Weak)

3.6.3.3 Strengths and Weakness of the Decision Tree approach

The strengths of decision tree methods are:

- Decision trees are able to generate understandable rules.
- Decision trees perform classification without requiring much computation.

- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

The weaknesses of decision tree methods :

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many classes and a relatively small number of training examples.
- The decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

3.6.3.4 Implementation

Python sklearn, DecisionTreeClassifier library was used as the classifier.

Kappa scores and accuracy were analyzed for all the three models to check which classifier was working best. Cohen's kappa coefficient (κ) is a statistic that is used to measure inter-rater reliability (and also Intra-rater reliability) for qualitative (categorical) items. It is generally thought to be a more robust measure than simple percent agreement calculation, as κ takes into account the possibility of the agreement occurring by chance.

3.7 Input abstracts to the classifier

SVM turned out to be the best classifier after analyzing the kappa scores and accuracy. Kappa scores and accuracy were analyzed for all the three models to check which classifier was working best. Cohen's kappa coefficient (κ) is a statistic that is used to measure inter-rater reliability (and also Intra-rater reliability) for qualitative (categorical) items. It is generally thought to be a more robust measure than simple percent agreement calculation, as κ takes into account the possibility of the agreement occurring by chance.
3 million abstracts were provided as input to the SVM classifier. Out of 3 mn abstracts, 1 mn abstracts were found to be relevant after classification. Further processing was done on these 1 mn abstracts.

3.8 Producing word embeddings

As a final stage, word embeddings were produced for the 1mn relevant abstracts by applying word2vec. A word embedding is a form of representing words and documents using a dense vector representation. The position of a word within the vector space is learned from the text and is based on the words that surround the word when it is used.

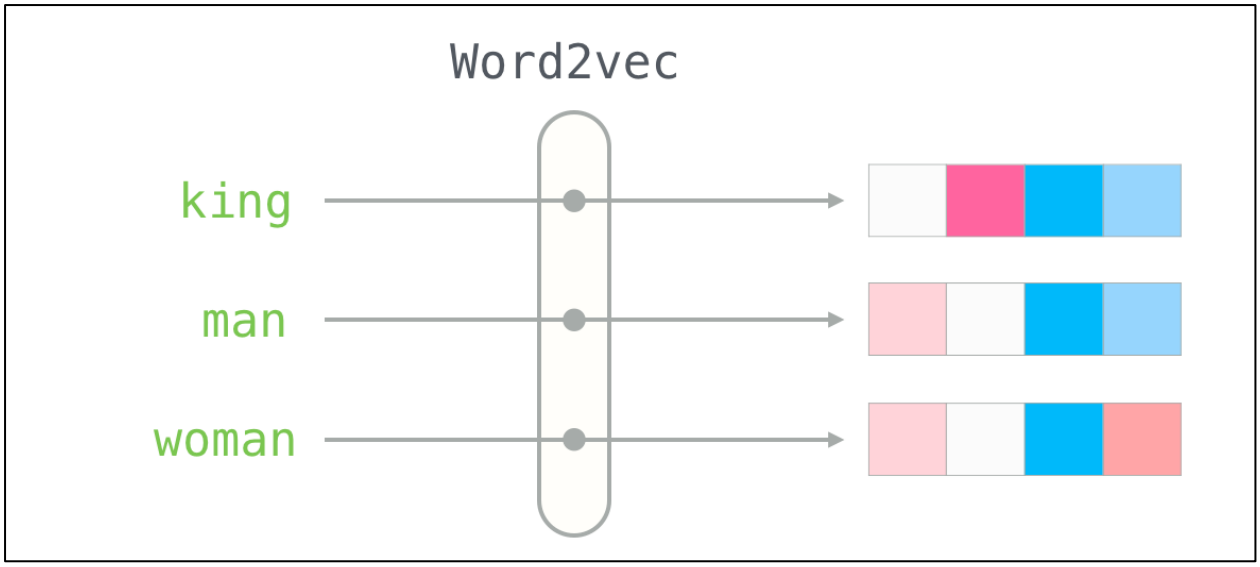

Figure 3.12 Representation of word2vec

Chapter 4

4 Results and Findings

4.1 Results for supervised classification

Classifier	Kappa score	Accuracy
1. Support Vector Machines	0.90	0.95
2. Logistic Regression	0.87	0.93
3. Decision Trees	0.70	0.85

Table 4.1 Kappa scores and accuracy for different classifiers

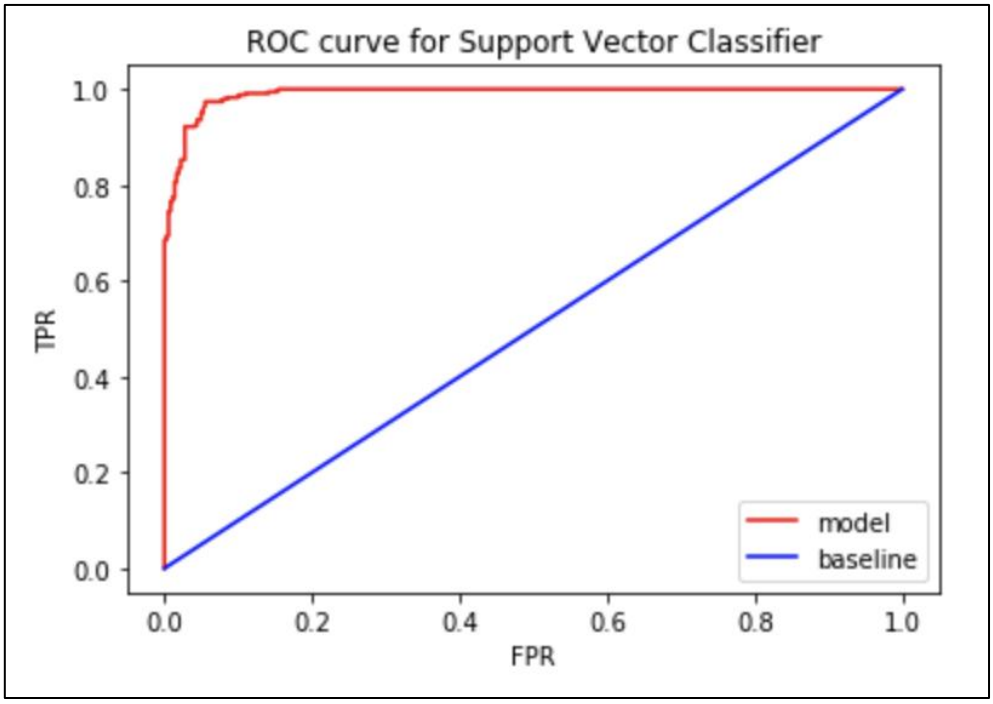

Figure 4.1 AUC-ROC curve for SVM classifier

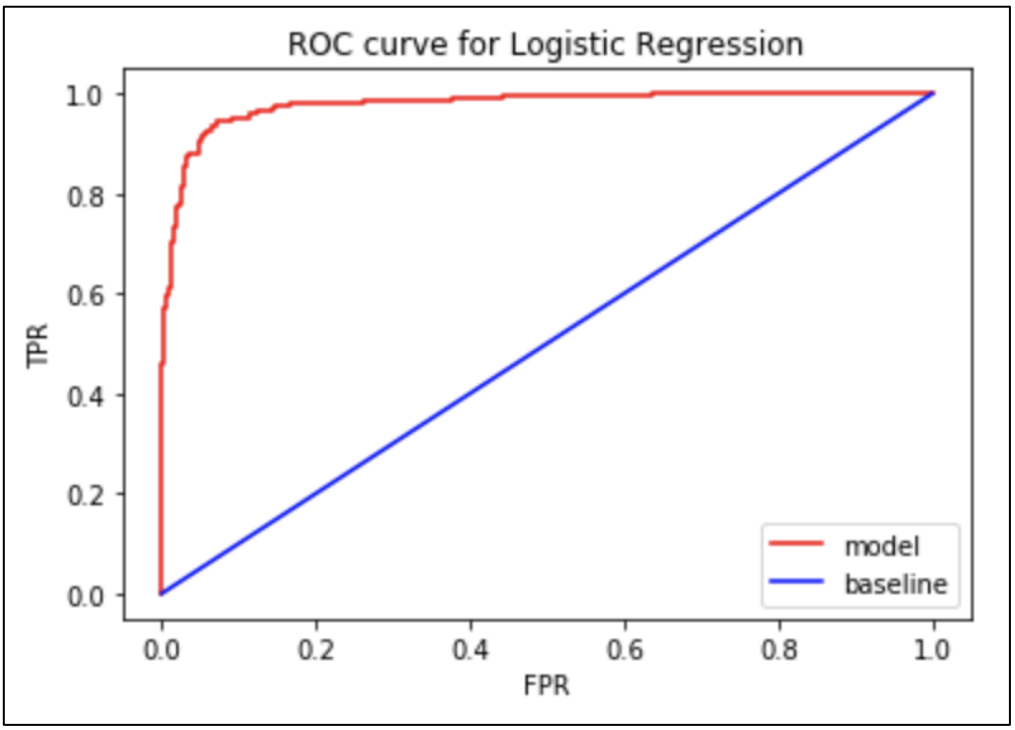

Figure 4.2 AUC-ROC curve for Logistic Regression classifier

According to the kappa scores and the accuracy, support vector machines (SVM) classifier was considered the best classifier for classifying the abstracts.

4.2 Results after applying word2vec

Word embeddings were produced for the 1 mn abstracts. We can analyze various gene-disease relationships on these embeddings and can produce some findings.
First, we carried out some basic experiments such as finding out similarities and analogies between the genes and the diseases to prove that our word2vec model worked well.
Then we tried to get some findings out of our model. Some of them were as follows:

4.2.1 Word Similarity

1. Most similar words to diabetes.

Result:

Similar words	Cosine similarity
Diabetic	0.7325131893157959
Diabete	0.7250828742980957
t2d	0.6969379782676697
t2dm	0.6920477151870728
t1dm	0.6192148923873901

Table 4.2 Most similar words to diabetes according to our word2vec model.

2. Most similar genes to diabetic gene TCF7L2

Results:

Similar genes	Cosine similarity
LEP	0.7408995628356934
IL12B	0.7376514673233032
FTO	0.7363024950027466
UGT1A1	0.7346640825271606
VDR	0.7346566915512085

Table 4.3 Most similar genes to the diabetic gene TCF7L2 according to our word2vec model.

4.2.2 Word Analogies

We also tried to find out answers to some questions using word analogies after applying our word2vec model. Some of these analogies are:

diabetes is to pancreas as asthma is to ?

To this, we got the most cosine similarity with the answer trachea. This is the best answer to our knowledge of the question asked.

Organ	Cosine similarity
trachea	0.55
bronchi	0.53
airways	0.50
bronchus	0.47
lungs	0.41

Table 4.4 Word Analogy for finding out similar organs

We could also determine the symptoms of a disease with the help of word analogy to our word2vec model. For example:
We asked the question

diabetes is to diuresis as asthma is to ?

To this, we got the best answer as ‘bronchoconstriction’ which is best to our knowledge.

Symptom	Cosine similarity
bronchoconstriction	0.48
asthmatic	0.47
cough	0.46
methacholine	0.45
asthm	0.44

Table 4.5 Word analogy to find out symptoms

4.2.3 Findings

We also tried to answer some questions that are new to our area of research. Some of these findings are:

4.2.3.1 Is MSH2 involved in breast cancer?

According to our model MSH2 has a cosine similarity of 0.46 to BRCA1 and should be treated as a breast cancer gene. But there is a dilemma in its role in breast cancer. In recent year's research papers it is said that MSH2 is not involved in breast cancer but in earlier papers, it was said that it is involved in breast cancer. So, this finding is crucial and needs to be verified.

4.2.3.2 MBL2 as a therapeutic strategy for lung cancer?

We found the most similar genes to immune checkpoint inhibitor gene CTLA4. According to our model, MBL2 showed the highest cosine similarity with CTLA4. MBL2, Mannose-binding lectin (MBL) is a recognition molecule that mediates phagocytosis and activates complement [11]. According to [12], the complement pathway can act as a therapeutic strategy for lung cancer. Recently, studies have demonstrated that intracellular activation of complement in cancer cells can act as an immunosuppressive pathway to regulate the expression of PD-L1. So there can be a future possibility that MBL2 can act as a therapeutic agent for cancer.

Chapter 5

5 Conclusion

We found out some useful gene-disease relationships by applying supervised classification and word embedding techniques. These techniques can also be used to carry out further findings in the area of research of biology. The key conclusions that we can draw out of our work are related to the functions of the MSH2 and MBL2 genes in breast cancer and therapeutics respectively. There is a dilemma in the role of MSH2 in breast cancer. In recent year's research papers it is said that MSH2 is not involved in breast cancer [13]. In earlier papers, it was said that it is involved in breast cancer [14], [15], [16]. So, this finding is crucial and needs to be verified. Complement pathway can act as a therapeutic strategy for lung cancer. Recently, studies have demonstrated that intracellular activation of complement in cancer cells can act as an immunosuppressive pathway to regulate expression of PD-L1 [12]. So there can be a future possibility that MBL2 can act as a therapeutic agent for cancer.

In conclusion, this work provides a stepping stone for numerous other dimensions of study in which different other gene-disease relationships can be examined, new theories can be formulated, and new mysteries can be solved.

References

[1] Mirończuk, M. M., & Protasiewicz, J. (2018). A recent overview of the state-of-the-art elements of text classification. *Expert Systems with Applications*, *106*, 36-54.

[2] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

[3] Pennington, J., Socher, R., & Manning, C. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

[4] Chiu, B., Crichton, G., Korhonen, A., & Pyysalo, S. (2016, August). How to train good word embeddings for biomedical NLP. In *Proceedings of the 15th workshop on biomedical natural language processing* (pp. 166-174).

[5] Manica, M., Mathis, R., Cadow, J., & Martínez, M. R. (2019). Context-specific interaction networks from vector representation of words. *Nature Machine Intelligence*, *1*(4), 181-190.

[6] Dernoncourt, F., & Lee, J. Y. (2017). Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. *arXiv preprint arXiv:1710.06071*.

[7] Zhang, Y., Chen, Q., Yang, Z., Lin, H., & Lu, Z. (2019). BioWordVec, improving biomedical word embeddings with subword information and MeSH. *Scientific data*, *6*(1), 52.

[8] Tshitoyan, V., Dagdelen, J., Weston, L., Dunn, A., Rong, Z., Kononova, O., ... & Jain, A. (2019). Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, *571*(7763), 95-98.

[9] https://bern.korea.ac.kr/

[10] https://github.com/ncbi-nlp/BioSentVec

[11] Wang, H. L., Lu, X., Yang, X., & Xu, N. (2016). Association of MBL2 exon1 polymorphisms with high-risk human papillomavirus infection and cervical cancers: a meta-analysis. *Archives of gynecology and obstetrics*, *294*(6), 1109-1116.

[12] Kleczko, E. K., Kwak, J. W., & Nemenoff, R. A. (2019). Targeting the complement pathway as a therapeutic strategy in lung cancer. *Frontiers in immunology*, *10*, 954.

[13] Maresca, Luisa, et al. "MSH2 role in BRCA1-driven tumorigenesis: a preliminary study in yeast and in human tumors from BRCA1-VUS carriers." *European journal of medical genetics* 58.10 (2015): 531-539.

[14] Wong, EE Ming, et al. "Is MSH2 a breast cancer susceptibility gene?." *Familial cancer* 7.2 (2008): 151-155. [14]

[15] Roberts, Maegan E., et al. "MSH6 and PMS2 germ-line pathogenic variants implicated in Lynch syndrome are associated with breast cancer." *Genetics in Medicine* 20.10 (2018): 1167. [15]

[16] Dowty, James G., et al. "Cancer Risks for MLH 1 and MSH 2 Mutation Carriers." *Human mutation* 34.3 (2013): 490-497. [16]