**Deep Learning Based Technique for Plagiarism Detection**

**ABSTRACT**

Plagiarism detection is very important especially for academician, researchers and students. Although, there are many plagiarism detection tools, it is still challenging task because of huge amount of online documents. In this research, we propose to use word2vec model to detect the semantic similarity between words in Arabic language which can help in detecting plagiarism. Word2vec is a deep learning technique that is used to represent words as features of vectors with high precision. The quality of vectors representation depends on the quality of corpus used in training phase. In this paper, we used OSAC corpus for training word2vec model. Moreover cosine similarity measure is used to compute the similarity between words’ vectors. The similarity measures show how simple changes in text such as changing one word, or changing the position of verbs and nouns results with similarity value equal to 99% which provide the possibility to detect plagiarism even if the test is altered by replacing words by their synonyms or changing the words order.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF SYMBOLS**  **LIST OF ABBREVIATIONS**  **LIST OF TABLES** | i  v  vii  xi  xii |
| 1. | **CHAPTER 1 : INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   3. EXISTING SYSTEM   1.3.1EXISTINGSYSTEM DISADVANTAGES  1.3.2 LITERATURE SURVEY  1.4 PROPOSED SYSTEM  1.4.1 PROPOSED SYSTEM ADVANTAGES |  |
| 2. | **CHAPTER 2 :PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION  2.2.3 MODULE DIAGRAM  2.2.4GIVEN INPUTAND EXPECTED OUTPUT  2.3 TECHNIQUE OR ALGORITHM |  |
| 3. | **CHAPTER 3 : REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS |  |
| 4. | **CHAPTER 4 : SYSTEM DESIGN**  **4.1 general**  4.1.1 activity diagram  4.1.2 USE CASE DIAGRAM  4.1.3 DATA FLOW DIAGRAM  4.1.4SEQUENCE DIAGRAM  4.1.5 COLLABORATION DIAGRAM  4.1.6CLASS DIAGRAM  4.1.7 SYSTEM ARCHITECTURE  4.1.8 OBJECT DIAGRAM  4.1.9 STATE DIAGRAM  4.1.10 COMPONENT DIAGRAM  4.1.11 E-R DIAGRAM  4.2 DATABASE DESIGN (ALL LEVEL |  |
| 5. | **CHAPTER 5 :SOFTWARE SPECIFICATION**  5.1 Python  5.2 History of python  5.3 impartance of python  5.4 features of python  5.5 libraries used |  |
| 6. | **CHAPTER 6 : IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION  6.3 DATA BASE TABLE STRUCTURE |  |
| 7. | **CHAPTER 7 : SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS |  |
| 8. | **CHAPTER 8 : SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING |  |
| 9. | **CHAPTER 9 :**  **APPLICATIONS AND FUTURE ENHANCEMENT**  9.1 GENERAL  9.2 APPLICATIONS  9.3 FUTURE ENHANCEMENTS |  |
| 10. | **CHAPTER 10 :**  10.1CONCLUSION  10.2 REFERENCES |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 2.3.2 | Module Diagram |  |
| 4.1 | Use case Diagram |  |
| 4.2 | Class Diagram |  |
| 4.3 | Object Diagram |  |
| 4.4 | Component Diagram |  |
| 4.5 | Deployment Diagram |  |
| 4.6 | Sequence diagram |  |
| 4.7 | Collaboration diagram |  |
| 4.8 | State Diagram |  |
| 4.9 | Activity Diagram |  |
| 4.10 | Data flow diagram |  |
| 4.11 | E-R Diagram |  |
| 4.12 | Architecture Diagram |  |

**LIST OF SYSMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class B  Class A  Class A  Class B | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processs. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | F inal state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Usecase |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or acion. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**LIST OF ABBREVATION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | ML | Machine Learning |
| 2. | SVM | Support Vector Machine |
| 3. | CNN | Convolutional Neural Networks |
| 4. | ANN | Artificial Neural Networks |
| 5. | AI | Artificial Intelligence |
| 6. | DNN | Deep Neural Networks |

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

Contextual word representation is very important for many Natural Language Processing (NLP) applications such as text classification, automatic summarization, information retrieval, query suggestions and plagiarism. Its importance related to the fact that it facilitates the process of finding relationships between two terms and computing their similarities. In order to compute the contextual representation of the words, word2vec method was used. Word2vec model is a deep learning technique that is used to compute the vector representation of words using neural network with one linear hidden layer on large dataset. In addition, word2vec train the model based on sliding window, the neighbor’s words within the window are taken into consideration to compute the probability of words occurrence, and moreover the window keeps sliding over the whole corpus recursively.

**1.2 OBJECTIVE**

The objective this project is to trains a model based on a neighborhood window of words in a corpus and then projects the result onto [an arbitrary number of] n dimensions where each word is a vector in the n dimensional space. Then the words can be compared using the cosine similarity of their vectors.

* 1. **Existing System:**

Contextual word representation is very important for many Natural Language Processing (NLP) applications such as text classification, automatic summarization, information retrieval, query suggestions and plagiarism. Its importance related to the fact that it facilitates the process of finding relationships between two terms and computing their similarities.

**LITERATURE SURVEY:**

**Title:** Arabic Text Classification Based on Word and Document Embedding

**Author:** Abdelkader El Mahdaouy

**Year:** 2017

**Description:**

Recently, Word Embedding have been introduced as a major breakthrough in Natural Language Processing (NLP) to learn viable representation of linguistic items based on contextual information or/and word co-occurrence. In this paper, we investigate Arabic document classification using Word and document embedding as representational basis rather than relying on text preprocessing and bag-of-words representation. We demonstrate that document Embedding outperform text preprocessing techniques either by learning them using Doc2Vec or averaging word vectors using a simple method for document Embedding construction. Moreover, the results show that the classification accuracy is less sensitive to word and document vectors learning parameters.

**Title:** A Survey of Attacks against Twitter Spam Detectors in an Adversarial Environment

**Author:** Niddal H. Imam

**Year:** 2017

**Description:**

Online Social Networks (OSNs), such as Facebook and Twitter, have become a very important part of many people’s daily lives. Unfortunately, the high popularity of these platforms makes them very attractive to spammers. Machine learning (ML) techniques have been widely used as a tool to address many cyber security application problems (such as spam and malware detection). However, most of the proposed approaches do not consider the presence of adversaries that target the defense mechanism itself. Adversaries can launch sophisticated attacks to undermine deployed spam detectors either during training or the prediction (test) phase. Not considering these adversarial activities at the design stage makes OSNs’ spam detectors vulnerable to a range of adversarial attacks. Thus, this paper surveys the attacks against Twitter spam detectors in an adversarial environment, and a general taxonomy of potential adversarial attacks is presented using common frameworks from the literature. Examples of adversarial activities on Twitter that were discovered after observing Arabic trending hashtags are discussed in detail. A new type of spam tweet (adversarial spam tweet), which can be used to undermine a deployed classifier, is examined. In addition, possible countermeasures that could increase the robustness of Twitter spam detectors to such attacks are investigated.

**Title:** Arabic Text Classification Based on Word and Document Embedding

**Author:** Abdelkader El Mahdaouy

**Year:** 2016

**Description:**

Recently, Word Embedding have been introduced as a major breakthrough in Natural Language Processing (NLP) to learn viable representation of linguistic items based on contextual information or/and word co-occurrence. In this paper, we investigate Arabic document classification using Word and document embedding as representational basis rather than relying on text preprocessing and bag-of-words representation. We demonstrate that document Embedding outperform text preprocessing techniques either by learning them using Doc2Vec or averaging word vectors using a simple method for document Embedding construction. Moreover, the results show that the classification accuracy is less sensitive to word and document vectors learning parameters.

**Title:** Word Embedding and Convolutional Neural Network for Arabic Sentiment Classification

**Author:** Abdelghani Dahou

**Year:** 2016

**Description:**

With the development and the advancement of social networks, forums, blogs and online sales, a growing number of Arabs are expressing their opinions on the web. In this paper, a scheme of Arabic sentiment classification, which evaluates and detects the sentiment polarity from Arabic reviews and Arabic social media, is studied. We investigated in several architectures to build a quality neural word embedding using a 3.4 billion words corpus from a collected 10 billion words web-crawled corpus. Moreover, a convolutional neural network trained on top of pre-trained Arabic word embedding is used for sentiment classification to evaluate the quality of these word embedding. The simulation results show that the proposed scheme outperforms the existed methods on 4 out of 5 balanced and unbalanced datasets.

**Title:** Character-Aware Neural Networks for Arabic Named Entity Recognition for Social Media

**Author:** Mourad Gridach

**Year:** 2016

**Description:**

Named Entity Recognition (NER) is the task of classifying or labelling atomic elements in the text into categories such as Person, Location or Organization. For Arabic language, recognizing named entities is a challenging task because of the complexity and the unique characteristics of this language. In addition, most of the previous work focuses on Modern Standard Arabic (MSA), however, recognizing named entities in social media is becoming more interesting these days. Dialectal Arabic (DA) and MSA are both used in social media, which is deemed as another challenging task. Most state-of-the-art Arabic NER systems count heavily on hand-crafted engineering features and lexicons which is time consuming. In this paper, we introduce a novel neural network architecture which benefits both from character-and word-level representations automatically, by using combination of bidirectional Long Short-Term Memory (LSTM) and Conditional Random Field (CRF), eliminating the need for most feature engineering. Moreover, our model relies on unsupervised word representations learned from unannotated corpora. Experimental results demonstrate that our model achieves state-of-the-art performance on publicly available benchmark for Arabic NER for social media and surpassing the previous system by a large margin.

**Proposed System**

Basically trains a model based on a neighborhood window of words in a corpus and then projects the result onto [an arbitrary number of] n dimensions where each word is a vector in the n dimensional space. Then the words can be compared using the cosine similarity of their vectors

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

Machine learning and data mining methods in predicting diseases were presented in the analysis Support Vector Machine (SVM) achieved a remarkable accuracy. Proposed an innovative technique which predicts the diabetes in people by identifying iris image using discrete wavelet transforms. Tried to increase the accuracy of the classifiers with an adaptive technique, employed improved k-means and logistic regression; the classifier achieved a better accuracy with a sufficient dataset.

**2.2 METHODOLOGIES**

**2.2.1** **MODULES NAME:**

1. Input File

2. Vector Representation

3. Level Treatment

4. Similarity Method

5. Plagiarism Detection

**1. Input File**

User need to provide the document to which user wants verify the plagiarism for. And the file will be taken as input to the plagiarism detector and it will be converted into vectors and then what is level of those vectors and then it will detect the plagiarism.

**2. Vector Representation**

This is a treatment performed on a text that will transform it to list of vectors which keep the semantic and syntactic aspect offered by the use of deep learning algorithms.

**3. Level Treatment**

This criterion defines the level of the treatment of a text, more exactly if the text is treated by word or by sentence.

**4. Similarity Method**

This part deals with the approaches used for calculating the similarity between the vectors that represent the texts, which will give us a global visibility to detect the strengths and weaknesses of each method. In addition, we are going to talk about the critical point for each approach illustrated in the paragraph above. Starting from the methods used for the vector representation of a text, according to the analysis it turns out that most of the approaches use either the word2vec or the doc2vec for its vector transformation, so we distinguish that the representations are the best methods used to keep the semantic aspect of a given text.

**5. Plagiarism Detection**

We found that almost of these approaches use the cosine to calculate the similarity between documents, so it was found that these methods perform its similarity analyses in word-by-word or sentence-by-sentence, which will pose after reliability problem of these results, since we can find two documents that share the same word or the same sentences but they are not semantically similar, in addition to that we can lose the semantic aspect when the documents are treating via a list of sentences or words.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

We can see from the results that on each database, the error rates are very low due to the discriminatory power of features and the regression capabilities of classifiers. Comparing the highest accuracies (corresponding to the lowest error rates) to those of previous works, our results are very competitive.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shouls what the system do and not how it should be implemented.

* PROCESSOR : DUAL CORE 2 DUOS.
* RAM : 4GB DD RAM
* HARD DISK : 250 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

**SOFTWARE REQUIREMENTS**

* Operating System : Windows 7/8/10
* Platform : Spyder3
* Programming Language : Python, HTML
* Front End : Spyder3

**3.4 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

**3.5 NON-FUNCTIONAL REQUIREMENTS**

**EFFICIENCY**

Our multi-modal event tracking and evolution framework is suitable for multimedia documents from various social media platforms, which can not only effectively capture their multi-modal topics, but also obtain the evolutionary trends of social events and generate effective event summary details over time. Our proposed mmETM model can exploit the multi-modal property of social event, which can effectively model social media documents including long text with related images and learn the correlations between textual and visual modalities to separate the visual-representative topics and non-visual-representative topics.

**CHAPTER 4**

**DESIGN ENGINEERING**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software.

**UML Diagrams**

Use Case diagram

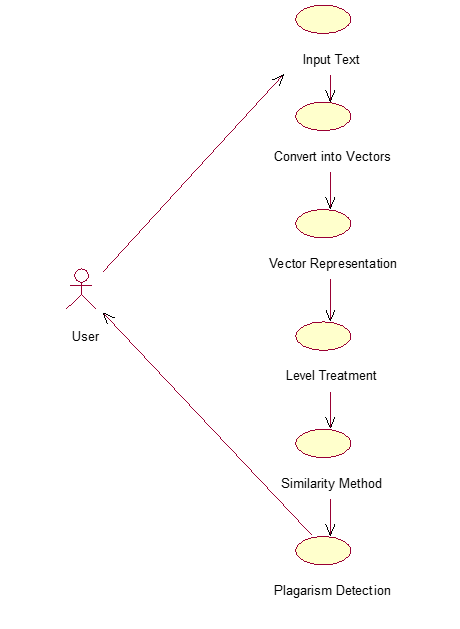


Fig 4.1: Use Case Diagram

**EXPLANATION:**

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

**Class Diagram**

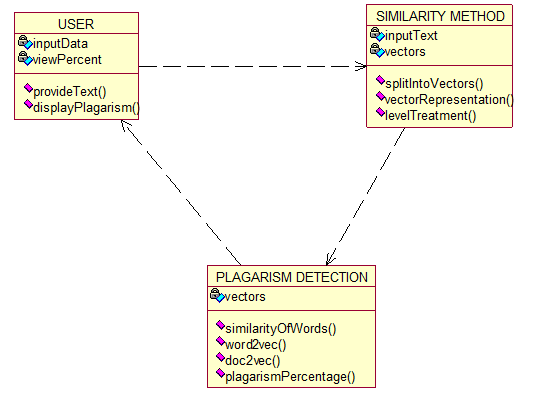


Fig 4.2: Class Diagram

**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

**Object Diagram**

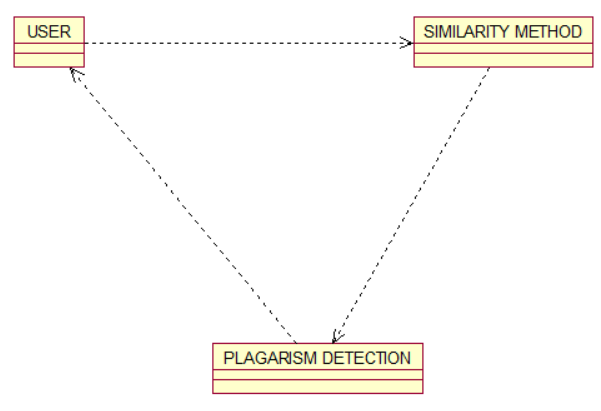


Fig 4.3: Object Diagram

**EXPLANATION:**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

**Component Diagram**

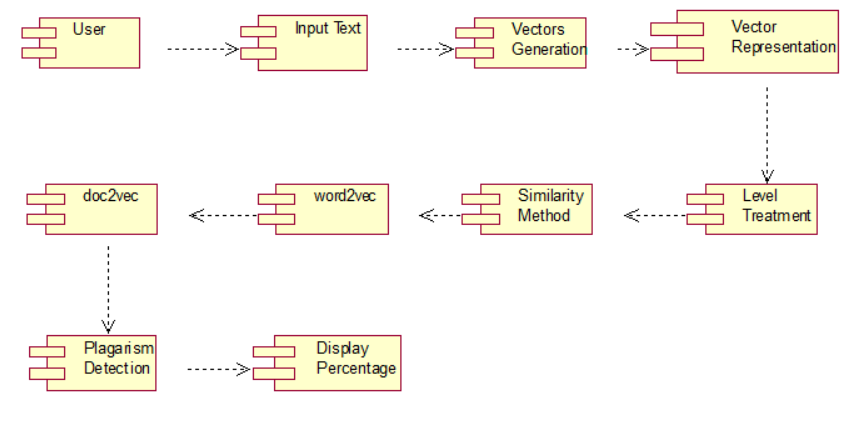


Fig 4.4: Component Diagram

**Deployment Diagram**

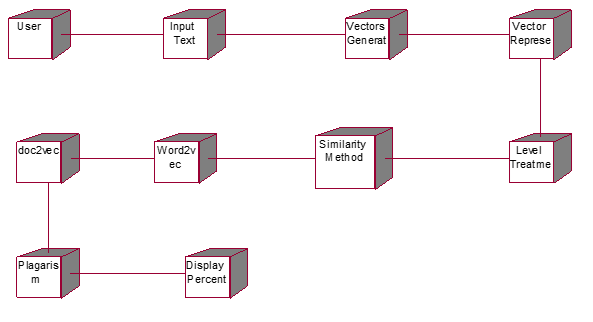


Fig 4.5: Deployment Diagram

**Sequence Diagram**

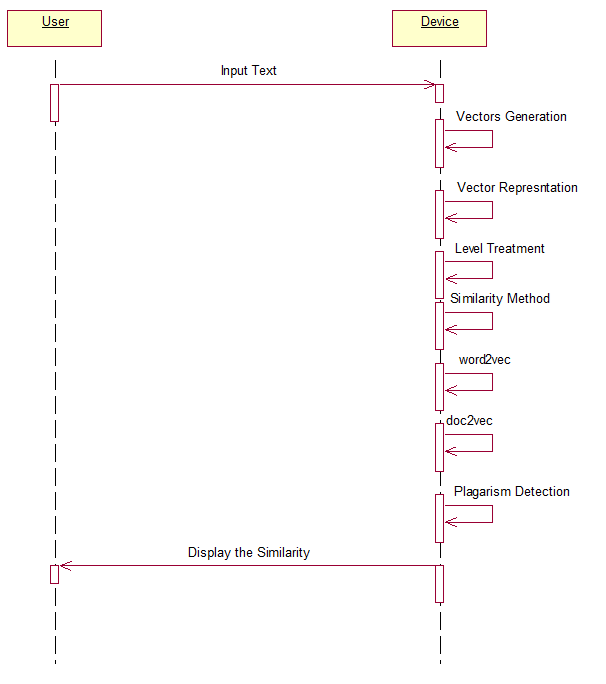


Fig 4.6: Sequence Diagram

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**Collaboration Diagram**

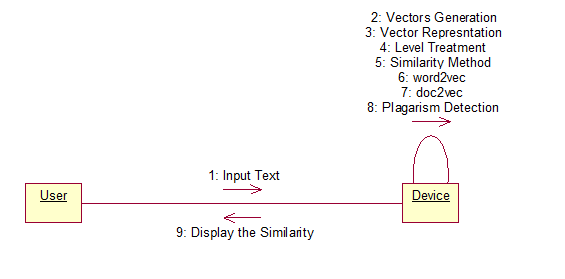


Fig 4.7: Collaboration Diagram

**State Diagram**

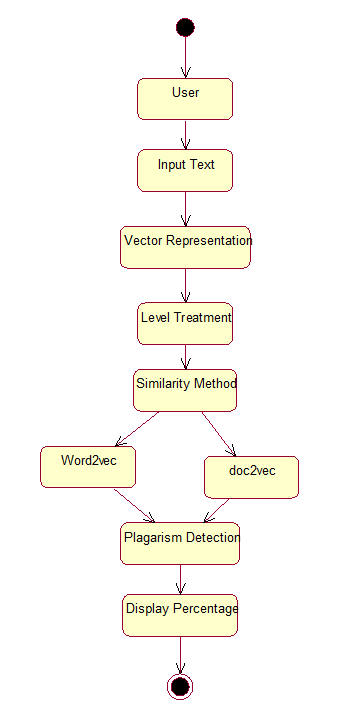


Fig 4.8: State Diagram

**EXPLANATION:**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**Activity Diagram**



Fig 4.9: Activity Diagram

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**Data Flow Diagram**

**Level 0**

Input Text

Vectors

Plagiarism Detection

Similarity

Word2vec, Doc2 vec

**Level 1**

Word2vec, Doc2vec

Input Text

Vectors

Display Percentage

Vector Generation, Level Treatment

Plagiarism Detection, Text Similarity, word or sentence

Similarity

Detection

Fig 4.10: Data Flow Diagrams

**EXPLANATION:**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

**E-R Diagram**

Vectors

Similarity

Display

User

Plagiarism

Fig 4.11: E-R Diagram

**EXPLANATION:**

Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database.

**System Architecture**

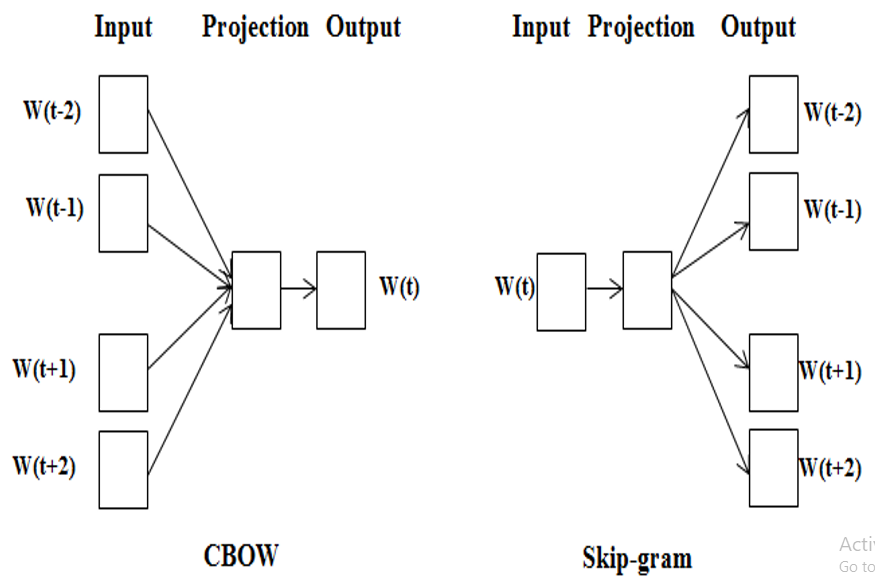


Fig 4.12: System Architecture

**CHAPTER 5**

**DEVELOPMENT TOOLS**

**5.1 Python**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

## 5.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

#### 5.3 Importance of Python

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### 5.4 Features of Python

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**5.5 Libraries used in python**

* numpy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as dataframes.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.



Figure : NumPy, Pandas, Matplotlib, Scikit-learn

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**Coding:**

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model\_selection import GridSearchCV

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, roc\_auc\_score, roc\_curve

# Importing the dataset

dataset = pd.read\_csv('../Dataset/diabetes.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 8].values

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 42)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Parameter evaluation

treeclf = DecisionTreeClassifier(random\_state=42)

parameters = {'max\_depth': [6, 7, 8, 9],

'min\_samples\_split': [2, 3, 4, 5],

'max\_features': [1, 2, 3, 4]

}

gridsearch=GridSearchCV(treeclf, parameters, cv=100, scoring='roc\_auc')

gridsearch.fit(X,y)

print(gridsearch.best\_params\_)

print(gridsearch.best\_score\_)

# Adjusting development threshold

tree = DecisionTreeClassifier(max\_depth = 6, max\_features = 4,

min\_samples\_split = 5,

random\_state=42)

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X, y, random\_state=42)

tree.fit(X\_train, y\_train)

print("Accuracy on training set: {:.3f}".format(tree.score(X\_train, y\_train)))

print("Accuracy on test set: {:.3f}".format(tree.score(X\_test, y\_test)))

# Predicting the Test set results

y\_pred = tree.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import classification\_report, confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print('TP - True Negative {}'.format(cm[0,0]))

print('FP - False Positive {}'.format(cm[0,1]))

print('FN - False Negative {}'.format(cm[1,0]))

print('TP - True Positive {}'.format(cm[1,1]))

print('Accuracy Rate: {}'.format(np.divide(np.sum([cm[0,0],cm[1,1]]),np.sum(cm))))

print('Misclassification Rate: {}'.format(np.divide(np.sum([cm[0,1],cm[1,0]]),np.sum(cm))))

round(roc\_auc\_score(y\_test,y\_pred),5)

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import accuracy\_score, roc\_auc\_score, roc\_curve

from sklearn.neighbors import KNeighborsClassifier

# Importing the dataset

dataset = pd.read\_csv('../Dataset/diabetes.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 8].values

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 42)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Parameter evaluation

knnclf = KNeighborsClassifier()

parameters={'n\_neighbors': range(1, 20)}

gridsearch=GridSearchCV(knnclf, parameters, cv=100, scoring='roc\_auc')

gridsearch.fit(X, y)

print(gridsearch.best\_params\_)

print(gridsearch.best\_score\_)

# Fitting K-NN to the Training set

knnClassifier = KNeighborsClassifier(n\_neighbors = 18)

knnClassifier.fit(X\_train, y\_train)

print('Accuracy of K-NN classifier on training set: {:.2f}'.format(knnClassifier.score(X\_train, y\_train)))

print('Accuracy of K-NN classifier on test set: {:.2f}'.format(knnClassifier.score(X\_test, y\_test)))

# Predicting the Test set results

y\_pred = knnClassifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import classification\_report, confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print('TP - True Negative {}'.format(cm[0,0]))

print('FP - False Positive {}'.format(cm[0,1]))

print('FN - False Negative {}'.format(cm[1,0]))

print('TP - True Positive {}'.format(cm[1,1]))

print('Accuracy Rate: {}'.format(np.divide(np.sum([cm[0,0],cm[1,1]]),np.sum(cm))))

print('Misclassification Rate: {}'.format(np.divide(np.sum([cm[0,1],cm[1,0]]),np.sum(cm))))

round(roc\_auc\_score(y\_test,y\_pred),5)

# Importing the libraries

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import accuracy\_score, roc\_auc\_score, roc\_curve

from sklearn.svm import SVC

import matplotlib.pyplot as plt

from sklearn.metrics import classification\_report

# Importing the dataset

dataset = pd.read\_csv('../Dataset/diabetes.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 8].values

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 42)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

#svm with grid search

svm = SVC(random\_state = 42)

parameters = {'kernel':('linear', 'rbf'), 'C':(1,0.25,0.5,0.75),

'gamma': (1,2,3,'auto'),'decision\_function\_shape':('ovo','ovr'),

'shrinking':(True,False)}

scores = ['precision', 'recall']

for score in scores:

print("# Tuning hyper-parameters for %s" % score)

print()

svm = GridSearchCV(SVC(), parameters, cv=5,

scoring='%s\_macro' % score)

svm.fit(X\_train, y\_train)

print("Best parameters set found on development set:")

print()

print(svm.best\_params\_)

print()

print("Grid scores on development set:")

print()

means = svm.cv\_results\_['mean\_test\_score']

stds = svm.cv\_results\_['std\_test\_score']

for mean, std, params in zip(means, stds, svm.cv\_results\_['params']):

print("%0.3f (+/-%0.03f) for %r"

% (mean, std \* 2, params))

print()

print("Detailed classification report:")

print()

print("The model is trained on the full development set.")

print("The scores are computed on the full evaluation set.")

print()

y\_true, y\_pred = y\_test, svm.predict(X\_test)

print(classification\_report(y\_true, y\_pred))

print()

svm\_model = SVC(kernel='rbf', C=100, gamma = 0.0001, random\_state=42)

svm\_model.fit(X\_train, y\_train)

spred = svm\_model.predict(X\_test)

print ('Accuracy with SVM {0}'.format(accuracy\_score(spred, y\_test) \* 100))

# Making the Confusion Matrix

from sklearn.metrics import classification\_report, confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print('TP - True Negative {}'.format(cm[0,0]))

print('FP - False Positive {}'.format(cm[0,1]))

print('FN - False Negative {}'.format(cm[1,0]))

print('TP - True Positive {}'.format(cm[1,1]))

print('Accuracy Rate: {}'.format(np.divide(np.sum([cm[0,0],cm[1,1]]),np.sum(cm))))

print('Misclassification Rate: {}'.format(np.divide(np.sum([cm[0,1],cm[1,0]]),np.sum(cm))))

svm.fit(X\_train, y\_train)

round(roc\_auc\_score(y\_test,y\_pred),5)

**CHAPTER 7**

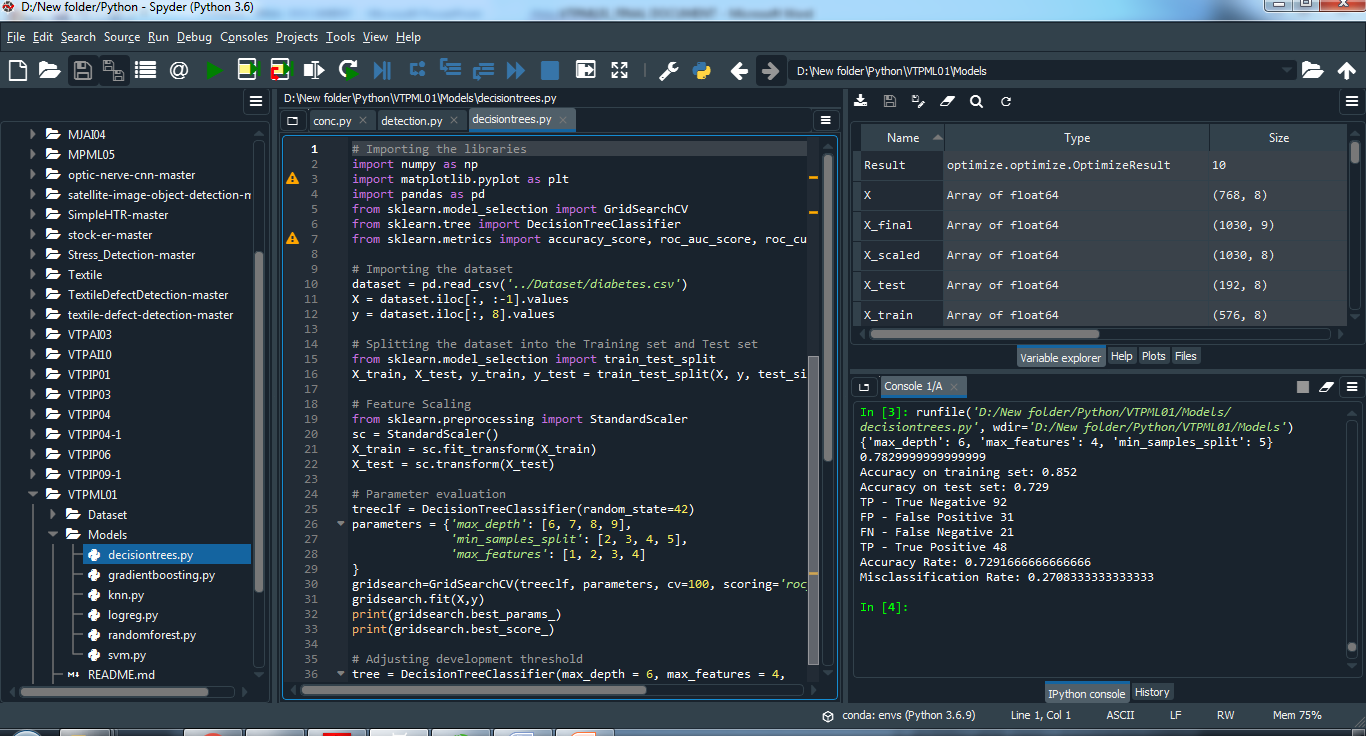
**SNAPSHOTS**

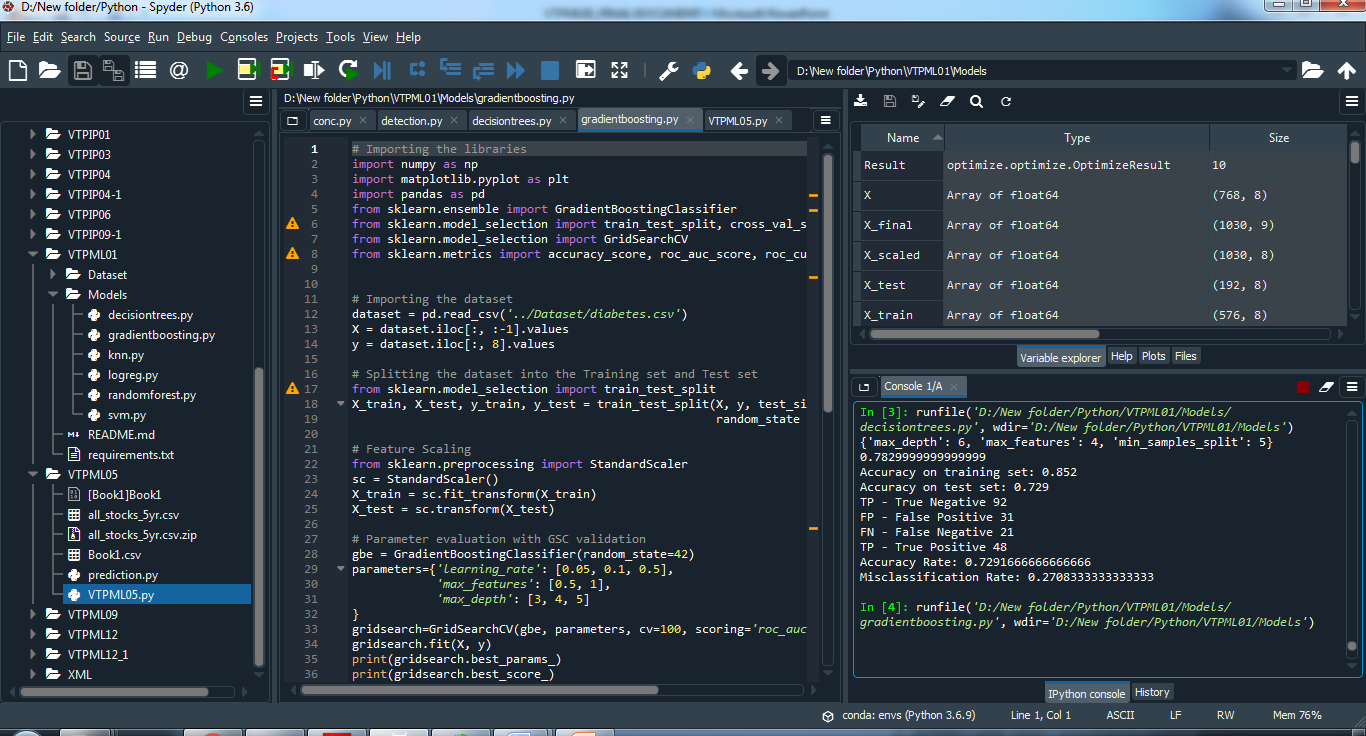
**General:**

This project is implements like application using python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

**SNAPSHOTS**



****

****

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3Types of Tests**

**8.3.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**8.3.2 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

**8.3.3 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**8.3.6 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**8.2.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**CHAPTER 9**

**FUTURE ENHANCEMENT**

An incremental decision tree algorithm is an online machine learning algorithm that outputs a decision tree. Machine learning tree classifiers other than the ones considered here can be analyzed in future for constructing hybrid tree classifier algorithms.

**CHAPTER 10**

**CONCLUSION & REFERENCE**

**10.1 CONCLUSION**

Plagiarism detection is one of serious tasks that represent a challenge for researchers, in this research we proposed to use word2vec model. However, Word2vec is a deep learning technique that uses large corpus for training, the output of this model is words that are represented as n dimensional vectors. Moreover, the cosine similarity between the vectors was used to detect plagiarism. In this case the similarity between vectors is contextual similarity since it depends on probability of occurrence of words within certain context. In addition to the fact that the quality of corpus determines the precision of vector representation which in turn affect the precision of plagiarism, in our experiments we used OSAC corpus. Therefore, our proposed technique is able to detect similarity between text if the changes are limited to single words replacement or order of verbs and nouns changed relatively. Accordingly, the experiment is able to detect plagiarism precision with 99% in this case.

**REFERENCES**

[1] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In ICLR.

[2] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large language models in machine translation. In Proceedings of the Joint Conference on Empirical Methods in Natural Language and Computational Language Learning, 2007.

[3] T. Mikolov, W.T. Yih, G. Zweig. Linguistic Regularities in Continuous Space Word Representations. NAACL HLT 2013

[4] Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137-1155, 2003.

[5] El Mahdaouy A., Gaussier E., El Alaoui S., 2016, Arabic Text Classification Based on Word and Document Embeddings, International Conference on Advanced Intelligent Systems and Informatics

[6] Bengio Y., Ducharme R., Vincent P., Jauvin C., A Neural Probabilistic Language Model, Journal of Machine Learning Research 3 (2003) 1137–1155

[7] Mikolov T., Sutskever I., Chen K., Distributed Representations of Words and Phrases and their Compositionality,

[8] T. Mikolov, V. Le Q., Sutskever I., Exploiting Similarities among Languages for Machine Translation, arXiv:1309.4168v1 [cs.CL] 17 Sep 2013

[9] Le Q., Mikolov T., Distributed Representations of Sentences and Documents, Proceedings of the 31 st International Conference on Machine Learning, Beijing, China, 2014. JMLR:W&CP volume 32

[10] Lin C., Ammar W., Levin C., Unsupervised POS Induction with Word Embedding, Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL, pages 1311– 1316,Denver, Colorado, May 31 – June 5, 2015.