**Plagiarism Detection**

**Major Project submitted in partial fulfilment of the**

**requirement for the award of degree of**

**MASTER OF TECHNOLOGY**

**in**

**Information Systems**

**Submitted By:**

**Akanksha Gupta**

**(University Roll No.02/IS/09)**

**Under the Guidance of**

**Prof. O.P.Verma**

**(HOD, IT Department)**

****

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**DELHI TECHNOLOGICAL UNIVERSITY**

**BAWANA ROAD, DELHI**

**(2009-2011)**

# CERTIFICATE

This is to certify that project entitled “**Plagiarism Detection**” has been completed by **Ms.Akanksha Gupta** in partial fulfilment of the requirement of **Master of Technology** in **Infomation Systems**.

The major project is a bonafide piece of work carried out and completed under my supervision and guidance during the academic session 2009-2011.The matter contained in this report has not been submitted elsewhere for the award of any other degree.This is a beneficial work in field of Plagiarism Detection which can further be used in any kind of searching in online documents using the upcoming Swarm Intelligence based technologies and resources.

**(Prof. O.P.Verma)**

**HOD & PROJECT GUIDE**

Dept. of Information Technology

**DELHI TECHNOLOGICAL UNIVERSITY**

BAWANA ROAD, DELHI – 110042

# ACKNOWLEDGEMENT

It is a great pleasure to have the opportunity to extend my heartiest felt gratitude to everybody who helped me throughout the course of this project. It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisors Prof. O.P.Verma , Ms. Seba Susan , Mr.Amioy Kumar and Mr. Anil Kumar Parihar for their invaluable guidance, encouragement and patient reviews.

This project has been a dedicated effort towards development of a very fast plagiarism detector using various approaches to optimise content searching, which primarily would not have been possible at the first place without the apt guidance of the Head of Information Technology Department, respected Prof.O.P.Verma. His motivation and encouragement during the project made me go further than expected in the development process and the redaction of this current report.

I would also like to take this opportunity to present my sincere regards to my teachers

for their support and encouragement.

This dedication would not be complete without expressing my gratitude towards all staff members and my dear classmates for their interest and curiosity toward my project.

Finally , I would also like to avail this opportunity to thank the IT Department and the college authorities for providing us all the necessary software and hardware requirements for the accomplishment of this project.

**(AKANKSHA GUPTA)**

**Master of Technology**

**(Information Systems)**

Dept. of Information Technology

**DELHI TECHNOLOGICAL UNIVERSITY**

BAWANA ROAD, DELHI – 110042

# Abstract

In thisproject we have improved the graph-based techniques of representing a document and accessing its constituent sentences. We have also proposed a new approach to reconstruct the evolution process in the database i.e.,the global corpus to optimize the no. of suspected texts. Further work has been done to extract tokens from the document and optimize the tokens to be used to detect the plagiarism based on graph representation. Tokens provide semantic metadata that summarize and characterize documents. We have also described an algorithm for automatically extracting keyphrases from text and providing a basis for optimized set of tokens for optimization techniques to be used like PSO , SA and BA .Our algorithm identifies candidate tokens using lexical methods, calculates local best, global best for each candidate token, and uses a machine-learning algorithm to predict which candidates are good tokens. The machine learning scheme first builds a prediction model using training documents , and then uses the model to find tokens in new documents.Preprocessing for each document is required such as breaking down the document into its constituent sentences. Segmentation of each sentence into separated terms and stop word removal.For the comparison of two documents graph is built by grouping each sentence terms in one node, The resulted nodes are connected to each other based on order of sentence within the document, All nodes in graph are also connected to top level node “Topic Signature”. Topic signature node is formed by extracting the concepts of each sentence terms and grouping them in such node. The topic signature is main entry for the graph is used as quick guide to the relevant nodes which should be considered for the comparison between source documents and suspected one. The proposed method has enabled us to achieve better performance in terms of effectiveness and efficiency of memory and CPU cycles used.

# 

# Motivation

As we all know in the recent years , in this fast growing web world , lot of plagiarism prevails today in digital texts , so it is very crucial to develop efficient and effective anti-plagiarism tools to prevent or identify document plagiarism[23].Also the earlier work in this field has been mainly into lexicographic methods or pairwise matching as in fingerprinting to detect the level of similarity between submitted and downloaded documents.

Graph based technique does provide a better way to represent online documents for searching and comparison to detect plagiarism . But lot scope in optimizing the no. of tokens used to compare and detect plagiarism was present. Maintaining less words/tokens in the topic signature of the graph corresponding to the document in examination , which aids in quick access to any part of the document helps to reduce a lot of memory space required to save pointers and arrays of sentence numbers attached to these tokens is reduced tremendously which another area of interest in today’s scenario where cost-cutting in the form of computation energy and memory used and speedy response in terms of searching in the web world is concerned are very high in priority .

Not just this , the usage and study of token optimization could also be used to automatically generate keywords for a document.To generate identifying keywords , first the set of optimized words can be generated and then manually , important and relevant words can be filtered out to characterize a document.

This behaves like optimizing the no. of buckets required to be maintained to store pointers to access any sentence in a document.

TABLE OF CONTENTS

[CERTIFICATE ii](#_Toc297241295)

[ACKNOWLEDGEMENT iii](#_Toc297241296)

[Abstract 1](#_Toc297241297)

[Motivation 2](#_Toc297241298)

[Chapter 1 : Plagiarism Detection 4](#_Toc297241299)

[1.1. Introduction 4](#_Toc297241300)

[1.2. Ontology Learning 7](#_Toc297241301)

[Chapter 2 : Related Work 9](#_Toc297241302)

[Chapter 3 11](#_Toc297241303)

[3.1. Optimization 11](#_Toc297241304)

[3.3. Types Of Optimization 13](#_Toc297241305)

[3.4. Algorithm Complexity 15](#_Toc297241306)

[3.5. Optimization Algorithms 17](#_Toc297241307)

[3.5.1. Bee Algorithms 18](#_Toc297241308)

[3.5.2. Simulated Annealing 21](#_Toc297241309)

[3.5.3. Partical Swarm Optimization 24](#_Toc297241310)

[Chapter 4 - Proposed Work 29](#_Toc297241311)

[4.1. Representation Of Online Document Using A Graph-Based Method 29](#_Toc297241312)

[4.2. Creation Of Global Corpus 34](#_Toc297241313)

[4.3. Application Of BA And Optimization Of The No. Of Tokens In The Topic Signature 36](#_Toc297241314)

[4.4. Application Of SA And Optimization Of The No. Of Tokens In The Topic Signature 38](#_Toc297241315)

[4.5. Our Approach Of Application Of PSO And Optimization 38](#_Toc297241316)

[Of The No. Of Tokens In The Topic Signature 38](#_Toc297241317)

[4.6. Peek into complete system 40](#_Toc297241318)

[Chapter – 6 : Experimental Study 42](#_Toc297241319)

[Conclusion And Future Work 47](#_Toc297241320)

[7.1. Conclusion 47](#_Toc297241321)

7.2 Future work………………………………………………………………………...48

[References 49](#_Toc297241322)

# 

# Chapter 1 : Plagiarism Detection

## Introduction

## DEFINITION : Plagiarism is defined as “using someone else’s words, ideas or results without attribution” [20].

PLAGIARISM is one of the forms of misuse of academic activities has increased rapidly in the quick and easy access to data and information through electronic documents and the Internet, and when we talk about plagiarism, we mean the text written by others where they are re-adjust the text to format by adding or deleting without any citation or reference.

Plagiarism can occur in two forms: (a) self-plagiarism, when authors reuse portions of their own previous writings in a subsequent paper [21], i.e., redundant and duplicate publications; and (b) “salami slicing”—i.e., dividing reports of the outcome of a research project into as many papers as possible in order to maximize the number of potential scientific publications [22].

Many approaches of plagiarism, such as copy and paste, which is the most common, redrafting or paraphrasing of the text, plagiarism of the idea, plagiarism through translation from one language to another and many other methods that use plagiarism are prevalent. Plagiarism is a serious problem in computer science. This is partly due to the ease with which electronic assignments may be copied,and to the difficulty in detecting similar assignments in a sufficiently large class. In addition, students are becoming more comfortable with cheating. A recent study found that 70% of students admit to some plagiarism, with about half being guilty of a serious cheating offense on a written assignment. In addition, 40% of students admit to using the “cut-and-paste” approach when completing their assignments .

This is not the only area where plagiarism is prevalent .Plagiarism has been found even in the area of research with many researchers and scientists using this means to provide extensions to their work by providing portions of previous works that are directly relevant to work at hand.[49].

Journalism is also another area where plagiarism is prevalent and has become a grievous problem. Since journalism's main currency is public trust, a reporter's failure to honestly acknowledge their sources undercuts a newspaper or television news show's integrity and undermines its credibility. Journalists accused of plagiarism are often suspended from their reporting tasks while the charges are being investigated by the news organization.[49]

The ease with which electronic text can be reproduced from online sources has lured a number of reporters into acts of plagiarism: Journalists have been caught "copying-and-pasting" articles and text from a number of websites.

PLAGIARISM DETECTION : Lot of plagiarism prevails today in the digital texts , so it is very crucial to develop efficient and effective anti-plagiarism tools to prevent or detect document plagiarism[23]. The key and main issue in plagiarism detection field is how to differentiate between plagiarized document and non-plagiarized document in effective and efficient way. Discovering Internet plagiarism in student papers has attracted a lot of attention from both researchers and software developers since the onset of the web phenomenon[24]. The first technique was **Fingerprint Matching**[25][26][27] which involves the process of scanning and examining the fingerprints of two documents in order to detect plagiarism. Then, **Clustering**[27][28]that uses specific words (or keywords) to find similar clusters between documents Fingerprinting techniques mostly rely on the use of K-grams[29] because the process of fingerprinting divides the document into grams of certain length k. Then, the fingerprints of two documents can be compared in order to detect plagiarism. It can, therefore, be classified fingerprints into three categories: character-based fingerprints, phrase-based fingerprints and statement-based fingerprints. The early fingerprinting technique uses sequence of characters to form the fingerprint for the whole document.

The current methods of plagiarism detection relay on the comparison of small text unit such as character, n-gram, chunk or terms to measure the similarity of documents. Suppose we have a document that contains ten sentences, each sentence contains five terms and each term consists of at least one character. The consideration of small text unit (character) for detecting of similarity between the original document and suspected document lead to a huge number of comparisons.

THE PLAGIARIST BEHAVIOUR **:** From observations it has been followed that the plagiarist behavior consists of some prototypical actions performed on the original text to obtain a new text. These prototypical actions are: insertion, deletion, substitution. It has also been observed that these actions can be performed at any level of the document structure, the plagiarist could for instance delete one paragraph from the document, insert one period in a paragraph, change one word in a period and so on[29].

Let s1 be a sub-sequence of chunks belonging to the same level L extracted from the original document. Let s2 be the sequence of chunks produced by the plagiarist starting from s1.Let a,b,c be chunks of the same level L.

Plagiarist may perform the following simple actions[29]:

* Insertion  
  The plagiarist may start from the original sequence s1 and insert a chunk.  
   s1=ab→s2=acb
* Deletion  
  The plagiarist may start from the original sequence ofchunks s1 and delete a chunk.  
   s1=acb→s2=ab
* Change  
  The plagiarist may start from the original sequence ofchunks s1 and substitute a chunk.  
   s1=acb→ s2=adb

## Ontology Learning

**Ontology learning** (**ontology extraction**, **ontology generation**, or **ontology acquisition**) is a subtask of information extraction. The goal of ontology learning is to (semi-)automatically extract relevant concepts and relations from a given corpus or other kinds of data sets to form an ontology.[48]

The automatic creation of ontologies is a task that involves many disciplines. Typically, the process starts by extracting terms and concepts or noun phrase from plain text using a method from terminology extraction. This usually involves linguistic processors (e.g. part of speech tagging, phrase chunking). Then statistical or symbolic techniques are used to extract relation signatures. For instance, these approaches try to detect that "to eat" denotes a relation between a concept denoted by "animal" and a concept denoted by "food". Recently, a graph-based approach has been proposed which extracts a domain taxonomy - i.e., the backbone of an ontology - from scratch[48].

Related areas of research to Ontology Learning are Semantic Web , Natural language Processing and Text Mining.

# 

# Chapter 2 : Related Work

The main domain of the plagiarism detection tools can be divided into two categories: text documents and program source codes[23]. A number of other dedicated Internet plagiarism detection software solutions have been developed in the last decade[24]. Some of the most popular solutions available on the market, are Turnitin [45], PAIRwise [43], Eve2 [41], MyDropBox [42], CopyCatch [40], and SafeAssign [44]. The first and the last services are the only two in the list that can be integrated with an existent university-wide course management system, but both of them require paid subscription. Table 1 shows plagiarism detection tools reported, the names of detection techniques,and their main domains of application[23].

**Table 1. Plagiarism detection tools in chronological order[23]. In Domain Column,** *text* **denotes plain texts and** *program* **denotes program sources.**

|  |  |
| --- | --- |
| Detection Tools | Domain |
| YAB[34] | Program |
| CHECK[33] | Text |
| Structural Character MOSS[35] | Program |
| SIM[36] | Program and Text |
| JPLAG[31] | Program |
| EVE2[37] | Text |
| SNITCH[30] | Text |
| PINT[38] | Program |
| Turn-it-in[30] | Text |
| Plague[38] | Program |

**Some Of The Available Plagiarism Detection Tools [12] :**

* [**www.plagiarism.org**](http://www.plagiarism.org/)
* [**www.urkund.com**](http://www.urkund.com/)
* [**www.copycatchgold.com**](http://www.copycatchgold.com/)
* [**www.plagiarism.phys.virginia.edu**](http://www.plagiarism.phys.virginia.edu/)
* [**www.canexus.com**](http://www.canexus.com/)
* [**http://www.plagiarism.com**](http://www.plagiarism.com/)
* [**www.cs.berkeley.edu**](http://www.cs.berkeley.edu/)
* [**www.ipd.uni‐karlsruhe.de**](http://www.ipd.uni‐karlsruhe.de/)

**All Kinds Of Literatures Plagiarism Data Chart Is Shown In Table 2.**

**Table 2 . ALL KINDS OF LITERATURES PLAGIARISM DATA CHART[39]**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Checked Amount | Average Percentage  Sim>80% | Plagiarism source  Similarity rate | Literature groups  repeat rate |
| Academic  papers | 167 | 13.2% | 3.3% | 0.4% |
| Curricular  papers | 212 | 79.6% | 26.4% | 4.5% |
| Master  thesis | 37 | 36.5% | 12.6% | 1.3% |
| Bachelor  Thesis | 72 | 67.4% | 19.6% | 2.1% |

# 

# Chapter 3

## Optimization

OPTIMIZATION is everywhere whether be to optimize CPU cycles needed to solve a problem or memory used to find a solution to a problem.

To reduce cost of infrastructure either in memory space or computation time all involve use optimization technique.

This is one example of need for optimization technique.Need for optimization techniques can be observed everywhere[1] from engineering design to financial markets,from our daily activity to planning our holidays, and computer sciences to industrial applications. We always intend to maximize or minimize something. An organization wants to maximize its profits, minimize costs, and maximize performance.Even in our day to day tasks , i.e.,daily chores we optimize cost of living to maximize benefits.

**HEURISTICS AND METAHEURISTICS**

In this paper, we propose a new method for plagiarism detection. The proposed method is graph-based, where each document is represented as graph. One node represents one sentence. Top level node is different node where it contains the concepts of terms in the document. Such node is called topic signature. The main advantage of proposed method is the topic signature which is the main entry for the

graph is used as quick guide to the relevant nodes, which should be considered for the comparison between source documents and suspected one.

Heuristics is a solution strategy by trial-and-error to produce acceptable solutions to a complex problem in a reasonably practical time. The complexity of the problem of interest makes it impossible to search every possible solution or combination, the aim is to find good, feasible solutions in an acceptable timescale [1].

* 1. **Engineering Optimization**

Optimization can include a wide range of problems with the aim of searching for certain optimality. Subsequently, there are many different ways of naming and classifying optimization problems, and typically the optimization techniques can also vary significantly from problem to problem. A unified approach is not possible, and the complexity of an optimization problem largely depends on the function forms of its objective functions and constraints.It is possible to write most optimization problems in the generic form mathematically

*fi(x),* (i = l,2,...,M)

subject to *φj(x)* = 0, *(j* = 1, 2,..., J )

*ψ(x)≤ 0 ,* (k = 1,2, . . . , K )

where *fi(x),* φj*(x)* and *ψ(x)* are functions of the design vector

*x = (x1,x2,...,xn)T.*

the components *xi* of *x* are called design or decision variables, and they can be real continuous, discrete or the mixed of these two. The functions *fi(x)* where *i =* 1,2, ...,M are called the objective functions, and in the case of *M* = 1, there is only a single objective. The space spanned by the decision variables is called the **design space** or **search space** Rn, while

the space formed by the objective function values is called the **solution space** or **response space**. The equalities for **φj*(x)***and inequalities for ***ψ(x)***are called **constraints**.

In a rare but extreme case where there is no objective at all, there are only constraints. Such a problem is called a **feasibility problem** because any feasible solution is an **optimal solution**.

## Types Of Optimization

Classification can be carried out in terms of the number of objectives, number of constraints, function forms, landscape of the objective functions, type of design variables, uncertainty in

values, and computational effort. The same can be described as in fig.3.1.

Fig.3.1

If we try to classify optimization problems according to the number of objectives,then there are two categories: single objective *M =* 1 and multiobjective *M* > 1. Multiobjective optimization is also referred to as multicriteria or even multi-attributes optimization in literature. Most real-world optimization problems are **multiobjective**. For example,when we rent or buy a house, we want it in the best possible location with a large space while we intend to spend as little as possible.

Similarly, we can also classify optimization in terms of number of constraints *J + K.* If there is no constraint at all *J = K =* 0, then it is called an **unconstrained optimization problem**. If *K =* 0 and *J >* 1, it is called an **equality-constrained problem**, while *J =* 0 and *K >* 1 becomes an **inequality-constrained problem**.

The actual function forms can also be used for classification. The objective functions can be either linear or nonlinear. If the constraints *<j)j* and *tpk* are all linear, then it becomes a linearly constrained problem. If both the constraints and the objective functions are all linear, it becomes a linear programming problem. Here 'programming' has nothing to do with computing programming; it means planning and/or optimization. Generally , if *fi(x),* φj*(x)* and *ψ(x)* are non-linear the the problem is a nonlinear optimization problem.

Optimization can also be classified in terms of the landscape of the objective functions. For a single objective function,the shape or the landscape may vary significantly for the nonlinear case. If there is only a single valley or peak with a unique global optimum, then the optimization problem is said to be **unimodal**. In this case, the local optimum is the only optimum, which is also its global optimum. For example, *f(x, y) = x2 + y2* is unimodal . However, most objective functions have more than one mode, and such multimodal functions are much more difficult to solve. For example, *f(x,y) =* sin(x) sin(y) is **multimodal**.

The type of value of the design variables can also be used for classification. If the values of all design variables are discrete, then the optimization is called **discrete optimization**. Discrete optimization is also referred to as the combinatorial optimization, though some literature prefers to state that the discrete optimization consists of integer programming

and combinatorial optimization. Combinatorial optimization is by far the most popular type of optimization and it is often linked with **graph theory** and **routing**, such as the **traveling salesman problem**, **minimum spanning tree problem**, **vehicle routing**, **airline scheduling** and **knapsack problem**.

If all the design variables are continuous or taking real values in some interval, the optimization is referred to as a **continuous** **optimization** problem. However, the design variables in many problems can be both discrete and continuous, we then call such optimization the **mixed type**. So far, the cases considered have exact values in both design variables and objective/constraints , and there is no uncertainty or noise in their values. All the optimization problems are **deterministic** in the sense that for any given set of design variables, the values of both objective functions and the constraint functions are determined exactly. In the real world, we can only know some parameters to a certain extent with uncertainty. If there is any uncertainty and noise in the design variables and the objective functions and/or constraints, then the optimization becomes a **stochastic optimization** problem, or a robust optimization problem with noise. For most stochastic problems, the function has to be redefined or reformulated in a way so that they become meaningful when using standard optimization techniques This often involves the averaging over some space and the objective should be evaluated in terms of mean and/or in combination with their related uncertainties.

## Algorithm Complexity

Computational complexity of an algorithm depends on the problem size. For example the sorting algorithm for a given number of *n* data entries, sorting these numbers into either ascending or descending order will take the computational time as a function of the problem size *n . 0(n)* means a linear complexity, while *0(n2)* has a quadratic complexity.From the viewpoint of computational effort associated with the optimization problem, we can classify them as NP and polynomial complexity. A solution to an optimization can be verifiable in polynomial time in a deterministic manner, often in terms of a determinist Turing machine, then we say the optimization is solvable in computational time or polynomial (P) time. This kind of problem has P-complexity. In this case,such a procedure or algorithm has P-complexity.

On the other hand, a problem is not solvable in polynomial time, we use the 'non-deterministic polynomial (NP) time' to refer to this case. Subsequently, the problem has the NP-complexity.And finding a solution to such a problem will take NP-time. In mathematical programming, an easy or tractable problem is a problem whose solution can be obtained by computer algorithms with a solution time (or number of steps) as a polynomial function of problem size *n.* Algorithms with polynomial-time are considered efficient. A problem is called the P-problem or polynomial-time problem if the number of steps needed to find the solution is bounded by a polynomial in *n* and it has at least one algorithm to solve it.

On the other hand, a hard or intractable problem requires solution time that is an exponential function of n, and thus exponential-time algorithms are considered inefficient. A problem is called nondeterministic polynomial (NP) if its solution can only be guessed and evaluated in polynomial time, and there is no known rule to make such guess (hence, nondeterministic). In

fact, no known efficient algorithms exist to solve NP-hard problems, and only approximate solutions or heuristic solutions are possible. Thus, heuristic and metaheuristic methods are very promising in obtaining approximate solutions or nearly optimal/suboptimal solutions.

A problem is called NP-complete if it is an NP-hard problem and all other problems in NP are reducible to it via certain reduction algorithms. The reduction algorithm has a polynomial time. The solvability of NP-complete problems (whether by polynomial time or not) is still an unsolved problem.

## Optimization Algorithms

For different types of optimization problems, we often have to use different optimization techniques because some algorithms are more suitable for certain types of optimization than others. In one extreme, suppose we are blind-folded without any guidance, so the search process is essentially a **pure random search**, which is usually not efficient. In another extreme, if we are told the highest peak of a known region, we will then directly climb up to the steepest cliff and try to reach to the highest peak, and this scenario corresponds to the classical **hill-climbing techniques**.

The search strategy can be refined a little bit further. Some hunters are better than others. We can only keep the better hunters and recruit new ones; this is similar to the **genetic algorithms** or **evolutionary algorithms** where the search agents are improving.

In general, optimization algorithms can be divided into two categories: **deterministic** **algorithms**, and **stochastic algorithms**. Furthermore, there is a third type of algorithm which is a mixture or hybrid of deterministic and stochastic algorithms called **hybrid algorithms**.

Most conventional or classic algorithms are **deterministic**. For example,the Simplex method in linear programming is deterministic. Some deterministic optimization algorithms used the gradient information, they are called **gradient-based algorithms**. For example, the well-known Newton-Rap hson algorithm is gradient-based as it uses the function values and their derivatives, and it works extremely well for smooth unimodal problems. However, if there is some discontinuity in the objective function, it does not work well. In this case, a non-gradient algorithm is preferred. Non-gradient-based or **gradient-free algorithms** do not use any derivative, but only the function values.

For stochastic algorithms, we have in general two types: **heuristic** and **metaheuristic**, though their difference is small. Loosely speaking, *heuristic* means 'to find' or 'to discover by trial and error'. Further development over the heuristic algorithms is the so-called **metaheuristic**

algorithms. Here *meta-* means 'beyond' or 'higher level', and they generally perform better than simple heuristics. In addition, all metaheuristic algorithms use certain tradeoff of randomization and local search. As such no agreed definitions of heuristics and metaheuristics exist in literature . The recent trends tend to name all stochastic algorithms with randomization and local search as metaheuristic.The classification can be summarized as shown fig.3.2.

Fig.3.2. Classification of algorithms

### Bee Algorithms

Bee algorithms form another class of algorithms which are closely related to the ant colony optimization. Bee algorithms are inspired by the foraging behaviour of honey bees. Several variants of bee algorithms have been formulated, including the Honey Bee Algorithm (HBA), the Virtual Bee Algorithm (VBA), the Artificial Bee Colony (ABC) optimization, the Honeybee Mating Algorithm (HBMA) and others.

The essence of the bee algorithms are the communication or broadcasting ability of a bee to some neighbourhood bees so they can 'know' and follow a bee to the best source, locations or routes to complete the optimization task.

***Behavior Of Honey Bees***

Honey bees live in a colony and they forage and store honey in their constructed colony. Honey bees can communicate by pheromone and 'waggle dance'. For example, an alarming bee may release a chemical message (pheromone) to stimulate attack response in other bees. Furthermore, when bees find a good food source and bring some nectar back to the hive, they will communicate the location of the food source by performing the so-called waggle dances as a signal system. Such signalling dances vary from species to species, however, they will try to recruit more bees by using directional dancing with varying strength so as to communicate the direction and distance of the found food resource. For multiple food sources such as flower patches, studies show that a bee colony seems to be able to allocate forager bees among different flower patches so as to maximize their total nectar intake. In order to survive the winter, a bee colony typically has to collect and store extra nectar, about 15 to 50 kg. The efficiency of nectar collection is consequently very important from the evolution point of view. Experimental studies have also been carried out by researchers, including the important work by S Camazine, T Seeley and J Sney in early 1990s and lately by Quijano and K. Passino and their colleagues.Various algorithms can be designed if we learn from the natural behaviour of bee colonies.

We elaborate a little more on the Bees algorithm and Artificial Bee Colony algorithms in the following paragraphs because they are relatively closer to our proposed algorithm.

***Bees Algorithm***

Pham et al., [2] proposed the Bees Algorithm which is a population-based algorithm that mimics the food foraging behaviour of swarms of honey bees. In its basic version, the algorithm performs a kind of static neighbourhood sampling combined with static random sampling. The Bees algorithm starts with the scout bees being placed randomly in the search space. The fitness of the scout bees is evaluated, after which, the bees that have the highest fitness are chosen as “selected bees” and sites visited by them are chosen for static neighbourhood sampling. The algorithm takes samples in the neighbourhood of the selected sites by placing bees in them, taking more samples near the best sites. For each site only the bee with the highest fitness is selected to form the next bee population. The remaining bees are used for random sampling. These steps are repeated until a stopping criterion is met.The same is as the pseudo code shown below .

**Pseudo Code Of Bee Algorithms**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Objective function f(x), x = ( x1 , ...,xn)T & constraints

Encode f(x) into virtual nectar levels

Define dance routine (strength, direction) or protocol

while ( criterion )

for loop over all n dimensions

(or nodes for routing and scheduling problems)

Generate new solutions

Evaluate the new solutions

end for

Communicate and update the optimal solution set

end while

Decode and output the best results

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***Artificial Bee Colony (ABC) Algorithm***

The ABC algorithm has been proposed by Karaboga [3], [4], [5], [6]. It is a population based algorithm. The colony consists of three groups of bees: employed bees, onlookers and scouts. A possible solution to the optimization problem is represented as the position of a food source and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The number of the employed bees is equal to the number of solutions in the population. At the first step, a randomly distributed initial population (food source positions) is generated. An employed bee produces a modification on the source position in its memory and evaluates the fitness at that position (calculates the nectar amount). If the fitness (nectar amount) of the new position is higher than that of the previous position, the bee memorizes the new source position and forgets the old one; otherwise it keeps the position of the old one in memory. After all the employed bees have evaluated the new positions, the onlookers go to these positions with more onlookers going towards better positions and less onlookers going towards less fit positions. The onlookers also produce a modification on that position and evaluate the fitness at that position. The scouts randomly select positions to evaluate. This cycle continues until the termination criteria is meet. Also if the fitness of certain employed bee does not improve for some time then that employed bee is converted to a scout bee.

### Simulated Annealing

Simulated annealing (SA) is a random search technique for global optimization problems, and it mimics the annealing process in material processing when a metal cools and freezes into a crystalline state with the minimum energy and larger crystal size so as to reduce the defects in metallic structures. The annealing process involves the careful control of temperature and cooling rate, often called annealing schedule.Simulated annealing algorithm is a search method using a Markov chain which converges under appropriate conditions concerning its transition probability.

***Markov Chain***

Most heuristic and metaheuristic search algorithms such as simulated annealing to be introduced later use a trajectory-based approach. They start with some initial (random) state, and propose a new state (solution) randomly. Then, the move is accepted or not depending on some probability. This is similar to a Markov chain. In fact, the standard simulated annealing is a random walk.

The Markov Chain Algorithm for Optimization is as shown in the fig. 3.3.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Objective function *f(x)*

Start with *U0 ϵ S,* at *t* = 0

while (criterion)

Generate *Yt+i* using an appropriate candidate kernel

Generate a random number 0 < *Pt <* 1

end

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Figure 3.3. The Ghate-Smith Markov chain algorithm for optimization

**Note** : Algorithms such as simulated annealing use a single Markov chain, which may not be very efficient. In practice, it is usually advantageous to use multiple Markov chains in parallel to increase the overall efficiency.

**Note :** It is worth pointing that this generic optimization framework may have good convergence under appropriate conditions. However, the computational efficiency is not always practical for large-scale problems.

The basic idea of the **simulated annealing algorithm** is to use random search in terms of a Markov chain, which not only accepts changes that improve the objective function, but also keeps some changes that are not ideal. In a minimization problem, for example, any better moves or changes that decrease the value of the objective function / will be accepted; however, some changes that increase / will also be accepted with a probability *p.* This probability *p,* also called the transition probability, is determined by

*p = e –*ΔΕ**/***kB T ,*

where *kB* is the Boltzmann’s constant, and for simplicity, we can use *k* to denote *kB* because *k=* 1 is often used. *T* is the temperature for controlling the annealing process. ΔΕis the change in energy levels. This transition probability is based on the Boltzmann distribution in physics.

The simplest way to link ΔΕwith the change of the objective function Δ*f* is to use

ΔΕ *= γ*Δ*f*

where *γ* is a real constant. For simplicity without losing generality, we can use *kB*=1 and *γ* =1. Thus, the probability *p* simply becomes

*p(*Δ*f,T) = e-*Δ*f/T*

Whether or not we accept a change, we usually use a random number *r* as a threshold. Thus, if *p > r* or

*p = e –* **Δ*f*/***T > r,*

the move is accepted.

The simulated annealing algorithm is as given in the fig.3.4.

Objective function *f(x), x=(x1,…,xp)T*

Initialize initial temperature T0 and initial guess *x(0)*

Set final temperature *Tf* and max number of iterations *N*

Define cooling schedule T *🡪 αT,* (0 < *α <* 1)

**while** ( *T > Tf* and *n< N )*

Move randomly to new locations: *xn+1=xn*+randn

Calculate Δ*f* = *f* n+1 ( x n+1 ) – *fn(xn)*

Accept the new solution if better

if not improved

Generate a random number *r*

Accept if *p =* exp [ - Δ*f* / T ] > *r*

**end** if

Update the best *x\** and *f*\*

*n= n+* 1

**end while**

Figure 3.4. Simulated annealing algorithm

* + 1. Partical Swarm Optimization

PSO is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995[8], inspired by social behavior of bird flocking or fish schooling. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called lbest. when a particle takes all the population as its topological neighbors, the best value is a global best and is called gbest.

The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its pbest and lbest locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pbest and lbest locations. The same is demonstrated as in fig.3.5.

In past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods.

Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

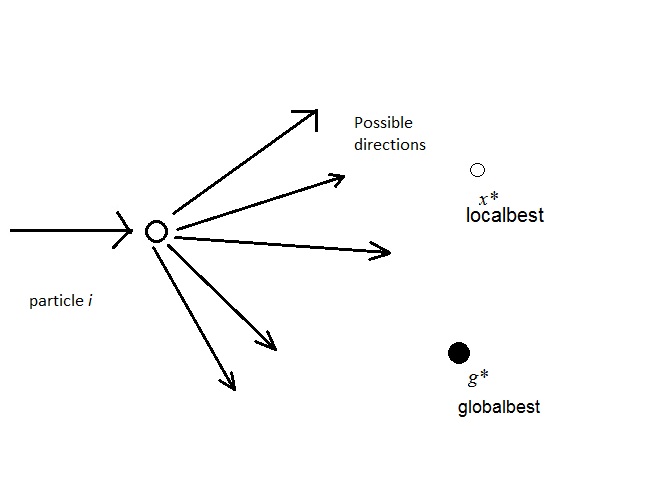


Figure . 3.5. Schematic representation of the motion of a particle in PSO, moving towards the global best *g\** and the current best *x\** for each particle *i.*

***The PSO Algorithm***

The PSO algorithm is guided by two factors:

a) Movement of the particle in local neighbourhood.

b) Movement of the particle in global neighbourhood.

Local Best Solutions: - are the best solutions due to particle itself searching for the best solution in the restricted swarm.

Global Best Solution: - is the best solution due to all the particles participating in the solution space.

Local and global best positions are updated only if better solution is found for each iteration.

Notations

i) Position of the ith particle:-

Xi = (Xi1, Xi2, ….., XiN) is the ith particle of the swarm. Here, the first subscript denotes the particle number and the second subscript denotes the dimension.

ii) Velocity of the ith particle:-

Vi =(Vi1, Vi2, ….., ViN)

iii) Local best position Xbest of the swarm:-

Xbest =(Pi1, Pi2, ….., PiN)

iv) Velocity Update:-

V*id (k+1)=(ω\**V*id k + γ1\*α(*Xbest *- Xid k)+ γ2\*β(gbest - Xid k)*

where : i = (1,2,……,m) is the number of swarms.

d= (1,2,…,N) is the dimension of the objective function to be optimized

gbest is the global best solution of the swarm.

k is the iteration number.

*ω* is the inertia weight to control the previous velocity vector of the swarm on the new one. It is a tradeoff between global and local exploration and helps in reducing the number of iterations for searching an optimal solution.

*γ1* and *γ2* are the random numbers between 0 and 1.

*α* is called the cognitive parameter.

*β* is called the social parameter.

Generally, *α* + *β* <=4 and by default, *α* = *β* =2.

v) Position Update:-

X*id (k+1)=* X*id + (* V*id (k+1) /q )*

where: q is the correction factor (optional) to speed up the convergence process.

The same can be summarized as pseudo code shown in figure 3.6.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Objective function *f(x), x = (x1, ...,xp)T*

Initialize locations *xi* and velocity *vi* of *n* particles.

Find *g\** from min{*f*(*x1*),..., *f(xn)}* (at *t =* 0)

**while** ( criterion )

*t = t +* 1 (pseudo time or iteration counter)

**for** loop over all *n* particles and all *p* dimensions

Generate new velocity υ*i t+1* using equation (15.1)

Calculate new locations *xi t+1* = *xt +* v*i* *t+1*

Evaluate objective functions at new locations *xt+1*

Find the current best for each particle *xi\**

**end for**

Fin the current global best *g\**

**end while**

Output the final results *xi\** and *g\**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Fig. 3.6. Pseudo code for Particle swarm optimization

## Graph Theory

DEFINITION : A graph G is a 4-tuple: G=(V,E,α,β), where V is a set of nodes (vertices), E ϵ V×V is a set of edges connecting the nodes, α : V → Σv is a function labeling the nodes, and β : V×V → Σe is a function labeling the edges (Σv and Σe being the sets of labels that can appear on the nodes and edges, respectively).

For brevity, we may refer to G as G = (V,E) by omitting the labeling functions. A graph

G1= (V1,E1,α1,β1) is a subgraph of a graph G2= (V2,E2,α2,β2), denoted G1 ⊆ G2, if V1 ⊆ V2 , E1 ⊆ E2 ϵ (V1 × V1) , α1(x) = α2(x) ∀ x ϵ V1 and β1(x, y) = β2(x, y) ∀ (x, y) ϵ E1. Conversely, graph G2 is also called a supergraph of G1.

# 

# Chapter 4 - Proposed Work

4.1. Representation Of Online Document Using A Graph-Based Method

In this part of implementation of plagiarism detector , we represented each document as a graph [12] .

Document conversion steps can be given as :

1) Select document for comparison.

2) Preprocessing of the document.

1. Convert the given document into compatible types. Eg. Convert pdf documents into doc files using http://www.pdftoword.com/ to make it compatible to be used with applications developed in Java for processing .
2. Add title tag , containing the name or the title of the document for aiding in document entry in database.

3) Convert document into graph.

1. Each unique term (word) appearing in the document, except for stop words such as “the”, “of”, and “and” which convey little information, becomes a vertex in the graph representing that document.

Terms/words/keywords can be extracted using any of the different methods available in [14][18][19].

1. Each node is labeled with the term it represents. We create only a single vertex for each word even if a word appears more than once in the text.
2. Further , these terms are grouped and linked to eachother in the order in which they come in a sentence and form the sentence node.
3. Every new word/term encountered is also linked to a separate list of words/terms called the topic signature which at all moments will maintain all the words/terms in a document. This topic signature will also be the entry point to reach to anyother part of the document in the graph form.
4. Whenever there is a new word/token/key[13] added to the topic signature , another list is maintained to contain links to all the sentence nodes which contain this word/token.

4) For comparison , the topic signature words/concepts[12]/token of the two documents

suspected and original are compared and on the instance of match , the corresponding list

of sentence nodes which contain this word node or vertex are referred . If a sentence has

already been detected to be similar[15],[16],[17] , based on the flag, the repetition is

omitted.

Datastructures used in graph representation are as below :

1. Term/Concept/Word node is given by

/\*\*

\* This class represents the node to hold the term

\* **@author** Akanksha Gupta

\*/

**class** DS\_term {

**private** String term;

**private** DS\_term next;

**private** List edge;

**public** String getTerm() {

**return** term;

}

**public** DS\_term getNext() {

**return** next;

}

**public** List getEdge() {

**return** edge;

}

**public** **void** setTerm(String term) {

**this**.term = term;

}

**public** **void** setNext(DS\_term next) {

**this**.next = next;

}

**public** **void** setEdge(List egde) {

**this**.edge = edge;

}

}

1. Sentence node is given by

/\*\*

\* This class represents the node to hold a Sentence

\* iSentenceNum : integer ,holds the sentence no. in the document

\* oTerms : List , holds the ordered sequence of terms contained in a

\* sentence

\* flag : booblean , to mark whether the sentence has been checked

\* before

\*

\* **@author** Akanksha Gupta

\*/

**class** DS\_node {

**int** iSentenceNum;

LinkedHashSet oTerms;

**boolean** flag = **false**;

**public** DS\_node() {

oTerms = **new** LinkedHashSet();

}

}

1. The node to represent the graph on a document is given by

/\*\*

\* This class is used to represent the total graph on the document

\* lsTopicSignature : Map , it contains the tokens as key paired with a

\* list having sentence numbers which contain this

\* token as a word in their sentence

\* document : List , contains the ordered list of sentence nodes as they

\* appear in the document

\* strDocTitle : String , contains the title of the document

\* **@author** mapsy

\*

\*/

**public** **class** DS\_graph {

Map lsTopicSignature;

List document;

String strDocTitle = **null**;

**public** DS\_graph() {

// **TODO** Auto-generated constructor stub

lsTopicSignature = **new** LinkedHashMap();

}

}

The sequence of token datastructures in a sentence list can be explained diagrammatically as shown in fig.4.1.

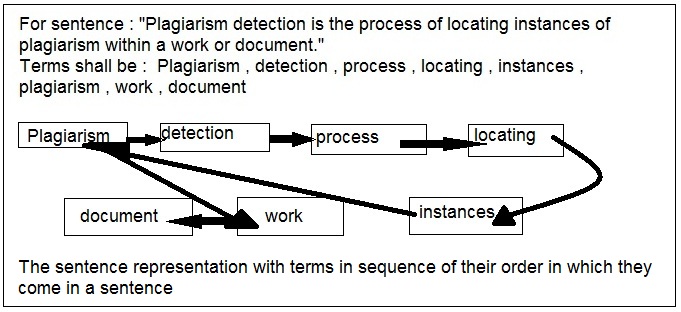


Fig.4.1. The sentence representation in a graph

The graph representation of a document can be diagrammatically explained as in fig.4.2.

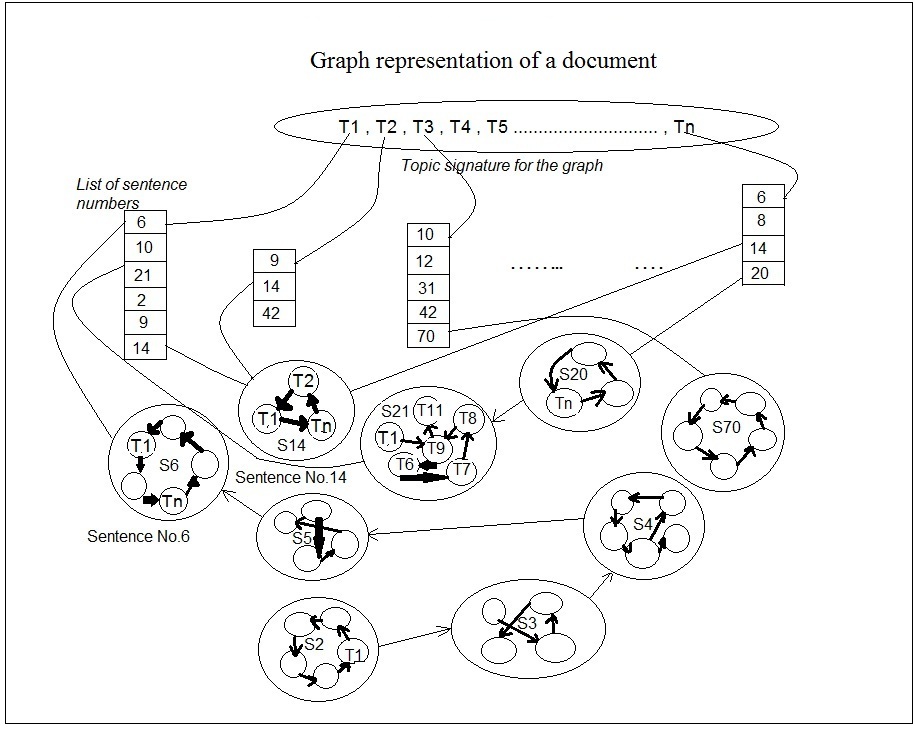


Fig.4.2 : Graph representation of a document

The comparison process can be diagrammatically explained as in fig.4.3.

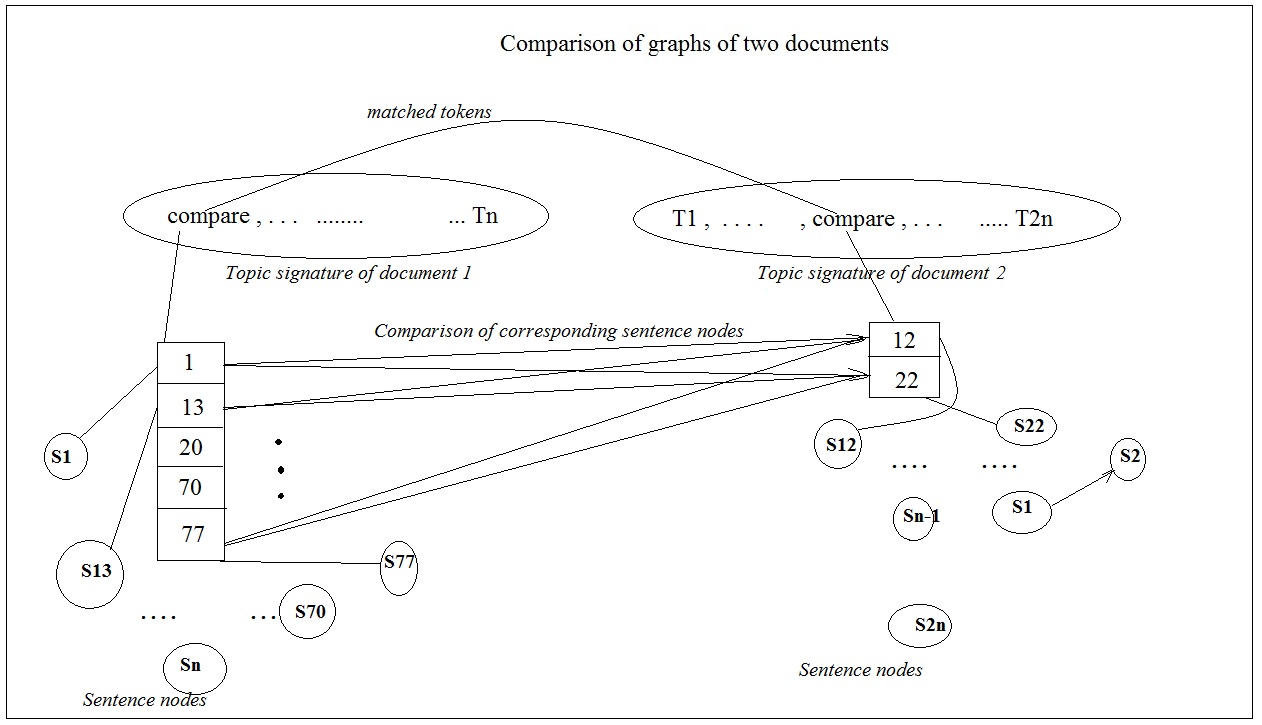


Fig.4.3 : Comparison to detect plagiarism using graph representation

## 4.2. Creation Of Global Corpus

Keyphrases or identifying words of a document provide semantic metadata that can be used to summarize or characterize documents[13].The extraction of the keyphrases is done using lexical methods , i.e., the document/file with text contents is read from left to right and top to bottom for keyphrases. The module for global corpus aids to identify the keyphrases from the tokens/words/terms obtained from previous module and calculates feature values for each of them.

Features used are as follows :

1. Localbest
2. Globalbest

­­­­All these values for each token/word obtained and its related document data is stored in a database for further processing of documents and their keyphrases.

Datastructures used in storing document and token/word related information in the database are as below :

1. To make an entry describing the document in the database

/\*\*

\* This class is used to make all the entries in the database table

\* document\_details

\* strDocumentTitle , String – used to hold the title of a document

\* iTotalNoOfConcepts , int – the total no. of concepts in the document

\*

\* **@author** Akanksha Gupta

\*/

**class TrainingDocumentEntry**

**{**

StringstrDocumentTitle**;**

intiTotalNoOfConcepts**;**

**}**

1. To make entries of concepts in a document in the database table document\_concept

/\*\*

\* This class is used to make all the concept entries of a document in the

\* database table document\_concept

\* strConcept , String – used to hold the concept

\* iNoOfOccurrences, int – the total no. of occurrences of the concepts in

\* the document

\* fLocalBest , float – the ratio of the no. of occurrences of the concept

\* to the total no. of concepts in the document

\*

\* **@author** Akanksha Gupta

\*/

**public** **class** DS\_ConceptLocal {

**public** String strConcept;

**public** **float** fLocalBest;

**public** **int** iNoOfOccurrences;

}

1. To ascertain the global value of the concept

/\*\*

\* This class is used to make concept entries in the database table

\* concept\_globalval

\* strConcept , String – used to hold the concept

\* iNoOfOccurrences, int – the total no. of occurrences of the concepts

\* in all the documents entered in the database

\* fGlobalBest , float – the ratio of the no. of occurrences of the

\* concept to the total no. of concepts in the database

\*

\* **@author** Akanksha Gupta

\*/

**public** **class** DS\_ConceptGlobal {

**public** String strConcept;

**public** **float** fGlobalBest;

**public** **int** iNoOfOccurrences;

}

The sequence of steps used in the creation of global corpus and the comparison of documents using optimized set of tokens/phrases can be explained diagrammatically as in fig.4.4 :

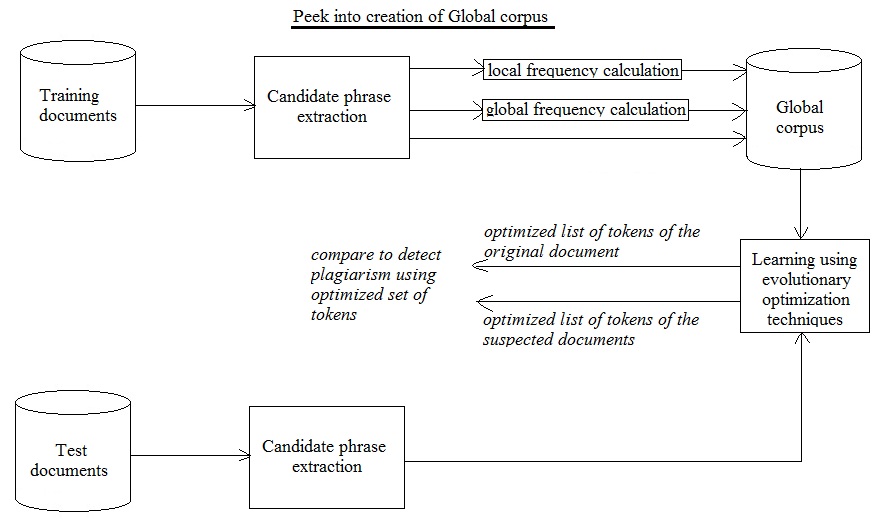


Fig.4.4 : Flow of extraction and calculation of candidate phrases and creation of global corpus

## 4.3. Application Of BA And Optimization Of The No. Of Tokens In The Topic Signature

Bee algorithms have many variants available for use.For our problem of optimizing the no. of tokens used to compare two documents , we have chosen ABC.The detailed implementation of the modified ABC to suit to our problem in hand is as pseudocode shown in fig. 4.5. The activity diagram of ABC is as shown in fig.4.6.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create a graph on a document under suspicion

Objective function *f(x), x = (x1,x2, ….. , xn)T*

Populate list with all the tokens of the topic signature and their corresponding

occurrences in the present document , call it *list1*

Sort the *list1* in decreasing order of the occurrences

while ( there are no items left in *list1*)

1. pick the highest occurrence token
2. check in topic signature for list of sentence numbers of this token
3. for(all token which have this token in their sentence nodes list.

* remove the sentence number from the list of the

of token call it a neighbourhood token , containing

this high priority token in its sentence node.

* if this neighbourhood token has an empty sentence

list , remove this token from the topic list.

* Update the optimal list (topic signature)
* Remove corresponding nodes from *list1*

end for

end while

Output optimized topic signature

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Fig.4.5. Pseudo code of ABC

The parameters values are as below :

1. The objective function taken as

Where *f(x) = original list of all tokens in the topic signature*

1. No . of *nectar levels* : n = no. of tokens in the original list of the topic signature
2. *Value of nectar level* of a token = count of occurrence of the token
3. Dance routine , *direction* = towards other tokens with this token in their sentence nodes , starting from the token with maximum occurrence
4. Iteration stopping *criterion* = until there are no more tokens left in the list
5. The *wi(j)* [1], the strength of the waggle dance of bee *i* at time step *t = j ,*

which decides probability of an observer bee following the dancing bee to forage

can be given by

*pi* = > 0 , is modified to just

where ,

= the total no. of sentences contained in the sentence list of this token *i* at

iteration *j .*

*nf* = total no. of tokens removed and tested so far

N (total no. of bees)= total no. of tokens in the original topic signature

The no. of observer bees hence at any iteration shall be equal to = N - *nf*

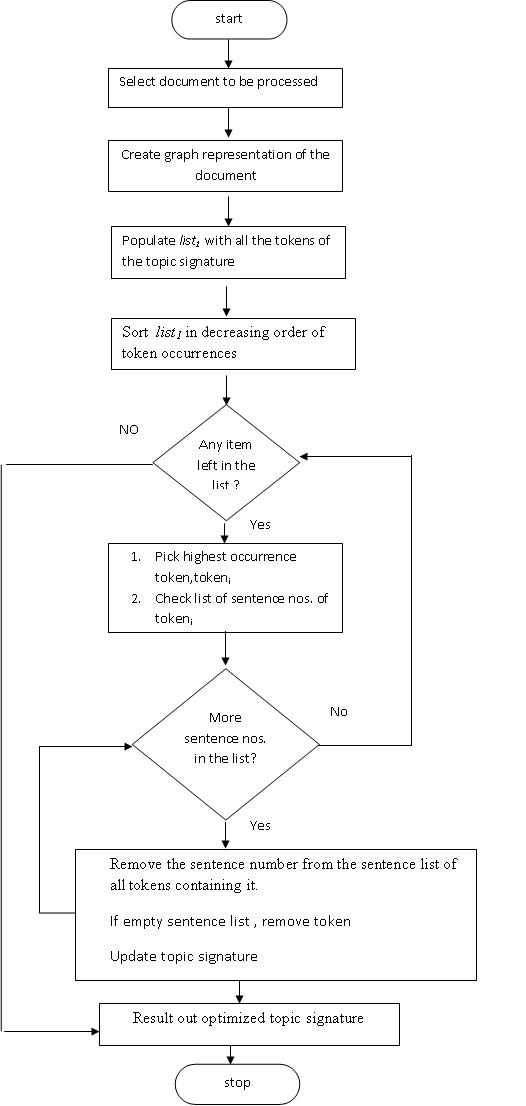


Fig.4.6. Flowchart of application of ABC to optimize topic signature

## 4.4. Our Approach Application Of SA And Optimization Of The No. Of Tokens In The Topic Signature

SA is a random search technique for global optimization problems , and it has been proven to converge to its global optimality. To reduce the number of tokens in the topic signature for comparison of documents and detect plagiarism , we have implemented the SA as in pseudo code shown in fig.4.7. and the flow chart of shown in fig. 4.8.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create a graph on a document under suspicion

Objective function min *f(token), token = (token1,token2, ….. , tokenn)T*

Initial Temperature T0 = N

Final Temperature Tf = 0

Populate list with all the tokens of the topic signature and their corresponding

occurrences in the present document , call it *list1*

N = total no. of tokens in *list1*

for i = 1 to N

repeat

Remove *tokeni* .

Check of 100% sentence coverage

if(sentence coverage == 100%)

update {topic signature} = {topic signature - *tokeni*}

end if

end for

Output optimized topic signature = min *f(token)*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Fig.4.7. Pseudo code of SA

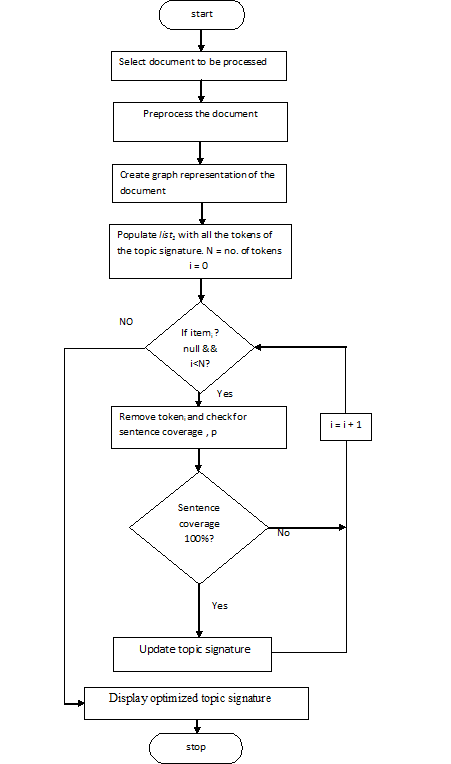
To improve the approach the list can be sorted in increasing order of the occurrences of tokens in the documents.

Parameters taken here are

1. Boltzmann constant kB = 1
2. *p* the transition probability is given by

*p* =

1. ∆f = *f(token) - fi(token),*
2. T = N = Total no. of tokens in the original topic signature
3. Random no. *r* =



Fi. 4.8. Flowchart of application of SA to optimize topic signature

## 4.5. Our Approach Of Application Of PSO And Optimization

## Of The No. Of Tokens In The Topic Signature

To identify keyphrases or optimize the number tokens present in the topic signature of the graph we created on a document , PSO was used as the learning algorithm for training and keyphrase extraction. In the creation of global corpus , we have made extensive use of data structures , Java Collections , JDBC and MySQL queries , sub-queries and PL/SQL. One by one as the tokens were entered in the database in the records meant for the respective document along their total no. of occurrences in the document so far and current local best value were also entered. The global table is updated simultaneously , with the global best value and inserted as a new row , if the token detected is not presented or the available row corresponding to the token and updated if the row is already presented. If the token is already available in the global table then , its global value is re-evaluated , and the total number of occurrences are increased with the additional number of occurrences.

Now , for the PSO application part , the velocity is initialized to zero and is updated with every new token added in the global corpus using the equation (derived from PSO):

1. The initial velocity for every token vtoken(0) = 0
2. Updated velocity for every token vtoken(t+1) = ω \*vtoken(t) + γ1 \* α \* (localbest – localbesttoken(t)) + γ2 \* β \* (globalbest - localbesttoken(t))

**PSO Parameters**

The initial values of the parameters of PSO are set as follows:

1) The number of swarms swarm\_size = total no. of tokens in the database

2) Maximum number of iterations itr = total no. of documents in the database

3) Inertia weight , ω = 1.0

4) Cognitive parameter , α = 2

5) Social parameter , β = 1

6) Parameters γ1 and γ2 (can be generated randomly in the interval [0-1]) , for this project

they are taken as = 0.5 and 0.5

In our approach , whenever we have a new document whose originality and credibility has to ascertained , we enter the tokens in its topic signature in the database and update their global and local best values.

Steps for comparison :

1. A matrix of these tokens is maintained to store them with their updated velocities which has a complexity of О(n) , where ‘n’ is the original no. of tokens in the topic signature.
2. To compare , a similar matrix is populated for the document chosen for comparison.
3. To search the database , all the lists of topic signature are compared with the current topic signature in consideration. Another list of common tokens in the two matched documents is created .
4. For all the matched tokens velocity difference is calculated in the decreasing order of velocities in the suspected document.
5. If the number of matched tokens is around 40% or more and the velocity differences between at least 5% of the matched tokens is less than 50% then the current original document could be a source of the copied document and the suspected document could be a case of plagiarism.
6. After cutting down of the original documents to be considered for matching , the actual matching between the documents is done using the lexical methods and graph representation as shown in fig.4.4.

The application of PSO to Global Corpus is as shown in Fig.4.9.

Select training document

start

Create graph representation of the document

Enter document details and tokens in the database

Update existing token frequencies and recalculate best values and velocities for all tokens

i = 0

stop

No

Yes

More training document?

Fig.4.9. Flowchart to show application of PSO to global corpus

## 4.6. Peek into complete system

Complete flow of execution of our proposed system can be understood as shown in the flowchart in fig.4.10. The document selected has to undergo some preprocessing before it could be used for graph representation , database entry and document / content comparison.

Preprocessing module has the following steps.

1. Convert into compatible document form.For the experiment , pdf documents were used and converted to word form using the pdftodoc.com .
2. A header containing the document title was manually inserted as a marker for the start of lexical analysis on the document.
3. Removal of white spaces , tab spaces , line-feed , etc.
4. Removal of unnecessary tokens like I , am , is , was , the , etc.

# 

# Fig.4.10. Flowchart of the proposed system

# Chapter – 5 : Experimental Study

The tokens of the topic signature were optimized using 3 different optimization algorithms , PSO , SA and BA. The comparison of the observation on running the algorithm on a no. documents in the database was carried out. The observation has been shown tabular form in Table 3.

**Table 3. Comparison of the three algorithms**

|  |  |  |  |
| --- | --- | --- | --- |
| **Optimization Algorithm Studied** | **Were all the sentence nodes navigated ?** | **Was plagiarism detection successful?** | **Was the entire global database searched ?** |
| PSO | No | Yes | Yes |
| SA | Yes | Yes | No |
| BA | Yes | Yes | No |

To demonstrate text breaking into tokens the example is as follows :

Let the text be :

My name is Akanksha. ----------------------------------------- Sentence 1

I stay in Delhi. ----------------------------------------------------Sentence 2

I am 5.7feet tall. --------------------------------------------------Sentence 3

I study at Delhi Technological University. -------------------Sentence 4

I love my name. --------------------------------------------------Sentence 5

Tokens : [name , Akanksha , stay , Delhi , 5.7feet , tall , study , Technological , Univeristy , love ]

Reduced Tokens PSO = [name , Delhi , Akanksha , stay , 5.7feet , tall , study , Technological , Univeristy , love ]

Reduced Tokens SA = [name , Delhi , 5.7feet ]

Reduced Tokens ABC = [name , Delhi , 5.7feet]

The observation obtained on running the algorithms to reduce tokens on different texts is as shown in Table.4.The code was run on paragraphs with 5 , 10 and 100 sentences .The tokens were obtained by creating the graph on the documents tested.

Table 4. The 3 paragraphs with 5 , 10 and 100 sentences

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm Used | No. of sentences tested in the para tested | Total no. of tokens identified | No. of tokens in the reduced token set | Total no. of tokens tested |
| BA | 5 | 11 | 4 | 4 |
|  | 10 | 100/74 | 5 | 5 |
|  | 100 | 1200 | 10 | 10 |
| SA | 5 | 11 | 4 | 11 |
|  | 10 | 100/74 | 5 | 100 |
|  | 100 | 1200 | 10 | 1200 |
| PSO | < 5 ~ 4 | 11 | 5-6 | 5-6 |
|  | < 10 ~ 7-8 | 100/74 | > 30 | > 30 |
|  | < 100 ~ 40 - 80 | 1200 | > 360 | > 360 |

In a 3-D cone depiction , when no. of sentences in the paragraph are 5 , the observation is as shown in fig.5.1.

Similarly , for no. of sentences 10 and 100 , the observation is as shown in fig.5.2. and fig.5.3. respectively.

Fig.5.1. 3-D cone representation when paragraph size is 5 sentences

Fig.5.2. 3-D cone representation when paragraph size is 10 sentences

Fig.5.3. 3-D cone representation when paragraph size is 100 sentences

Fig.5.4. The cluster cylinder representation for no. of sentences = 5 .

Fig.5.5. The cluster cylinder representation for no. of sentences = 10.

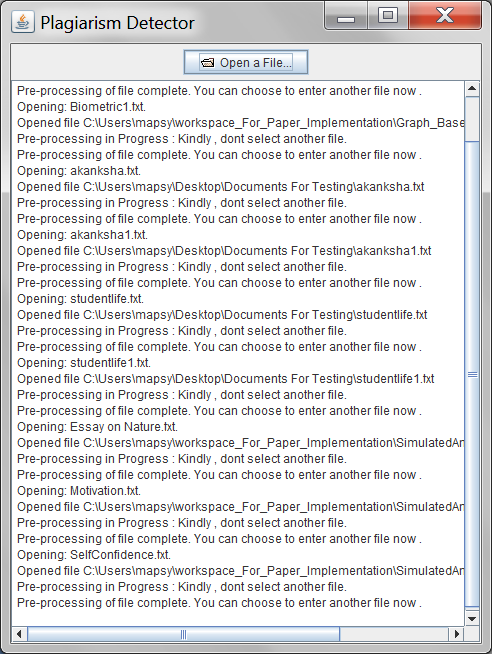
Fig.5.6. The cluster cylinder representation for no. of sentences = 100 .

The trend of applying the three different optimization algorithm to the problem of optimizing the no. of tokens in the topic signature , can be clearly seen in the fig.5.7. Here , we can have plotted the observation of applying the 3 algorithms to the three different kinds of documents one with 5 lines , another with 10 lines and third with 100 lines. The graph clearly shows , that ABC and SA optimize efficiently which becomes even more noticeable as the size of text increases. PSO definitely gives optimized results but ABC and SA have proved to be better for this problem.

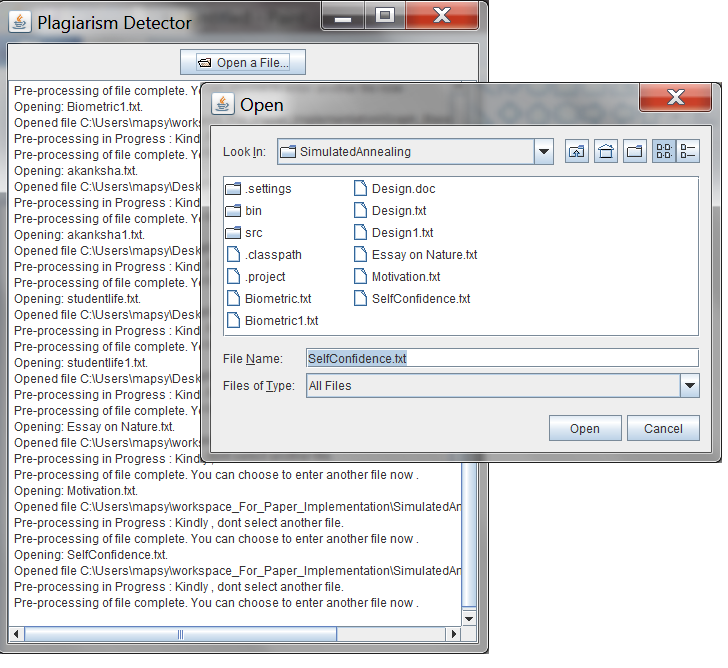
Fig.5.7. The complete trend for no. of sentences = 5 , 10 , 100 .

The proposed system as we implemented as explained in the chapter 4 has been shown using the screenshots of the created system.

1. The training data creation/entry is done using the UI shown in figure 5.8. It is simple swing based UI that uses a JFileChooser Java class . The same has been shown in figure 5.9.

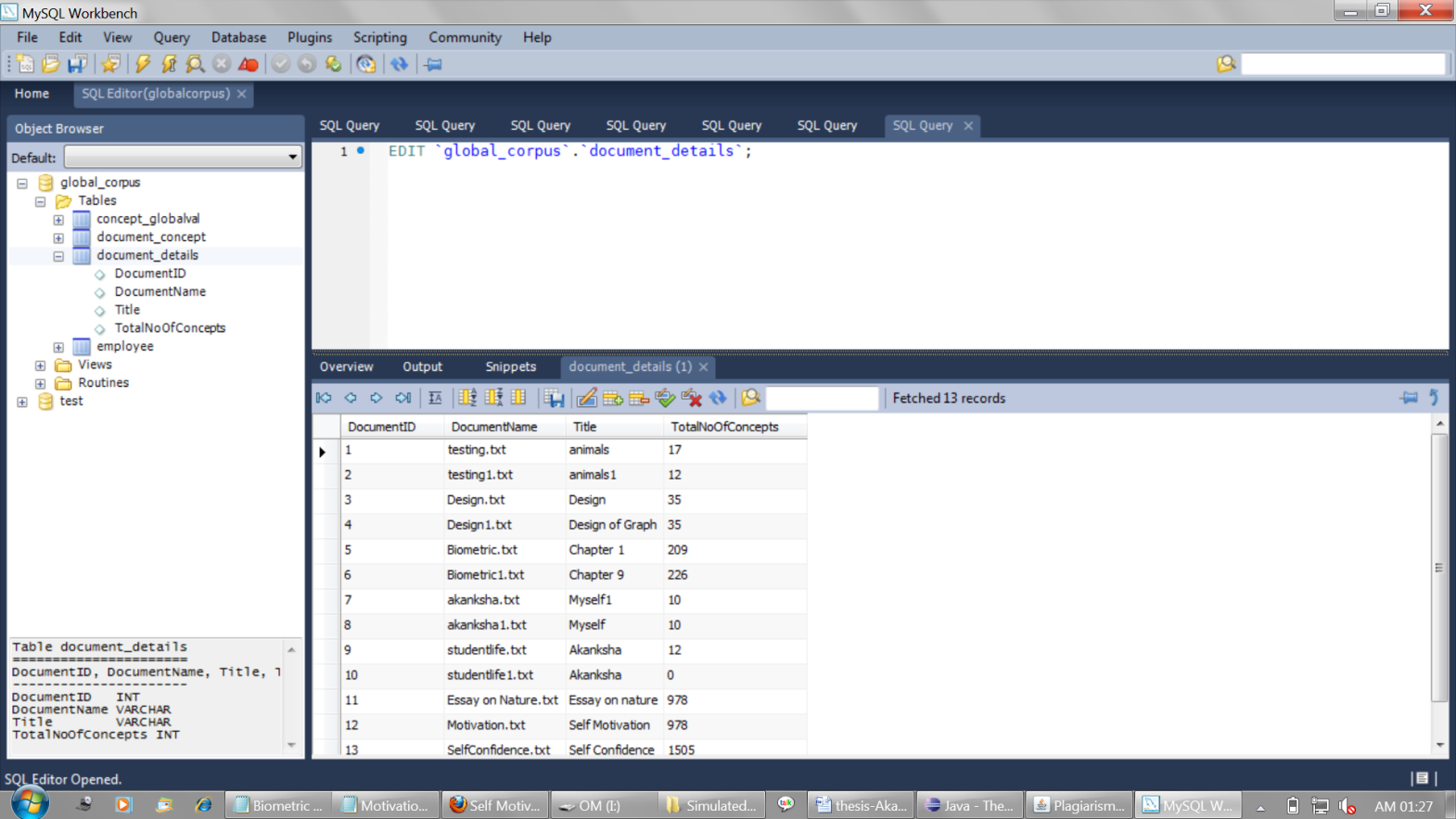
****

**Fig.5.8. Training data entry UI**

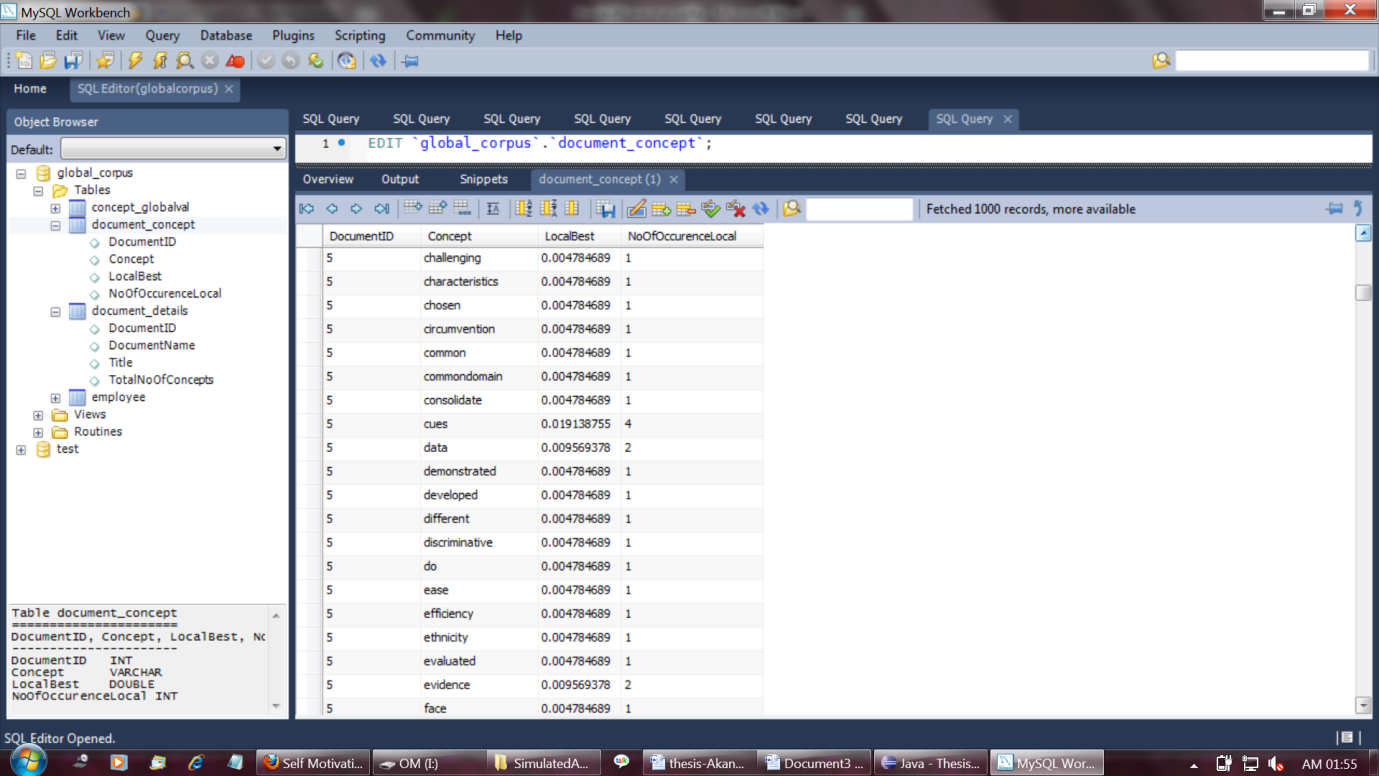
****

**Fig.5.9 : The training data entry UI and data entry in process as can be seen in the panel**

1. The training data entry into the database is shown in fig.5.10. In this figure we can see the document details getting persisted in the database. Further tokens details getting saved in the database can be seen in fig.5.11.

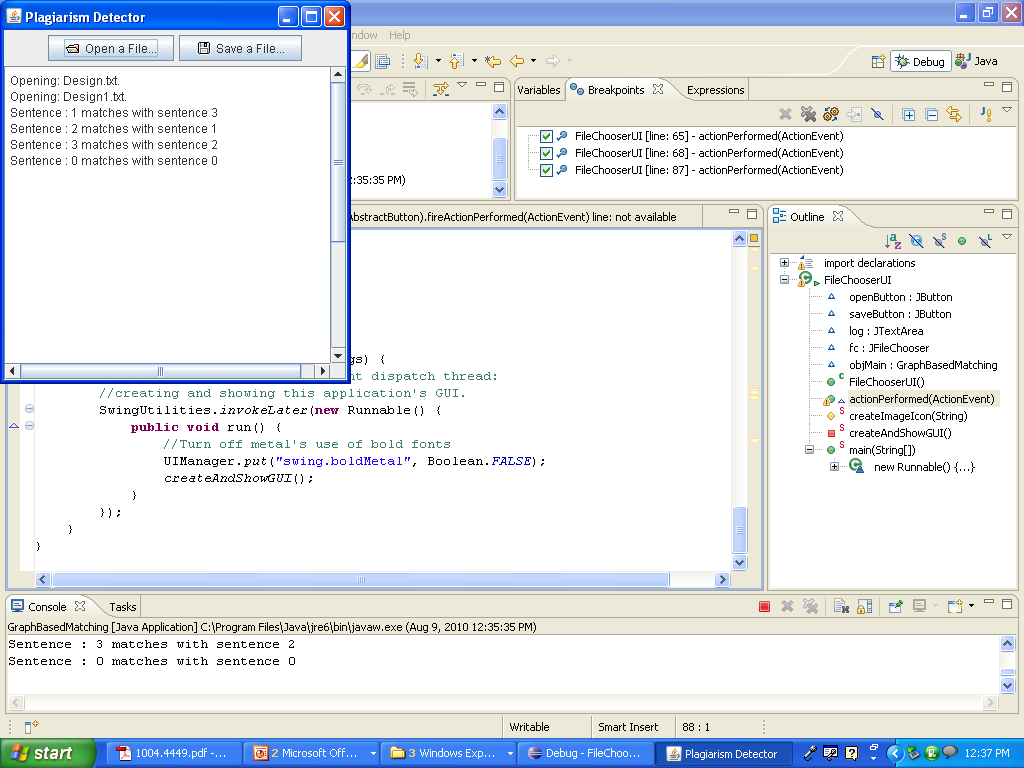
****

**Fig.5.10. Document details entry in the database.**

****

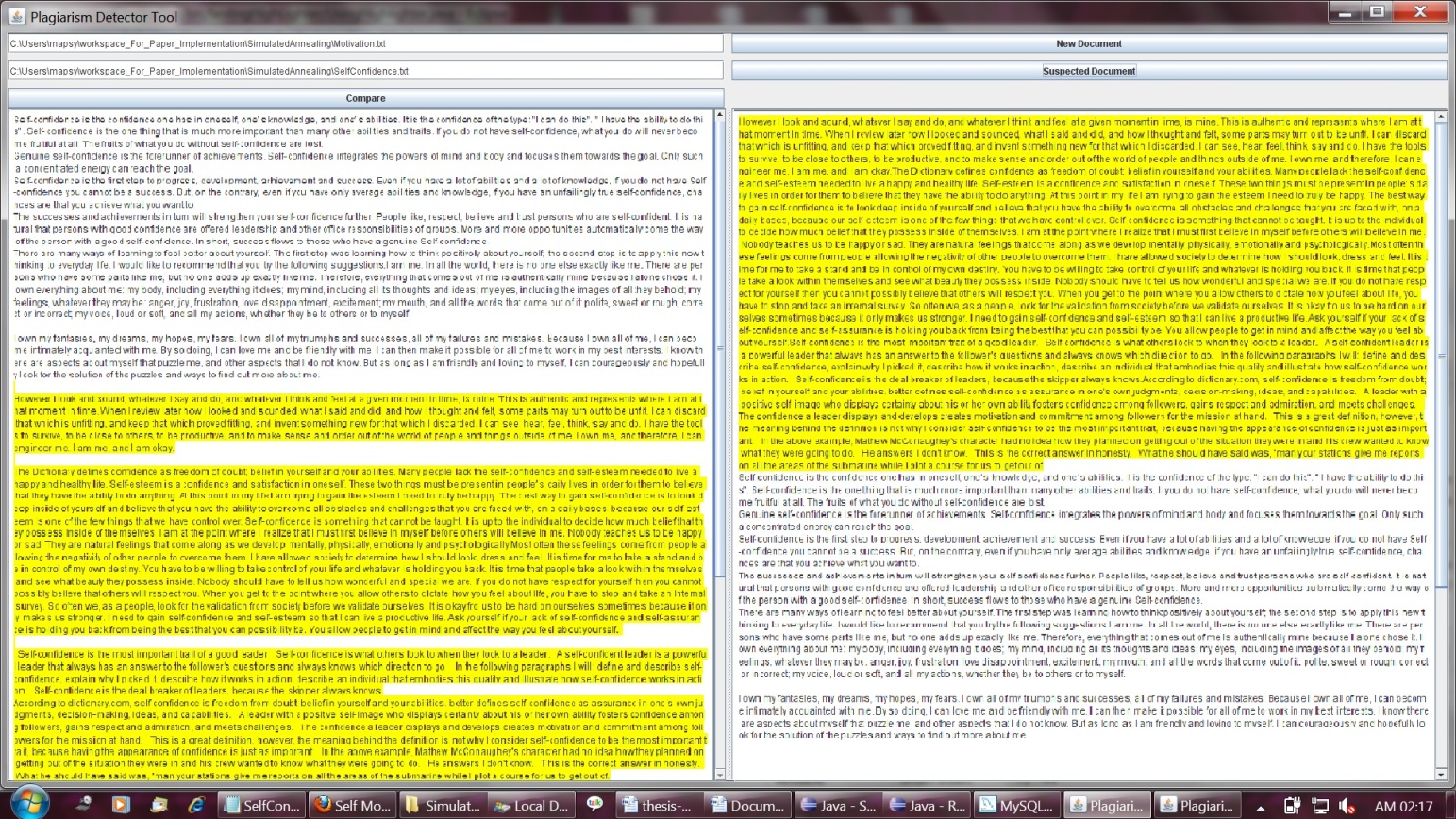
**Fig.5.11. Token entry in the database.**

1. The plagiarism detection in documents using simple lexicographic methods as implemented by us and the same displayed on Java Eclipse console is shown in fig.5.12.

****

**Fig.5.12.Plagiarism Detection using lexicographic methods and console display**

1. Plagiarism detection with optimized tokens using SA/ABC/PSO on the topics signature and suspected document search in database optimized using PSO was implemented and the following fig.5.13 shows how the proposed system looks like. The documents compared some copied paragraphs. The display was created using JPanel and JTextArea and the copied / plagiarized paragraphs detected were displayed in the textarea as highlighted text using HighLighter class in Java.

****

**Fig.5.13. The plagiarism detection tool implemented using the proposed method**

# 

# Conclusion and Future Work

## Conclusion

Deleting and tracing text plagiarism is an important issue , especially in the internet space and where knowledge credibility is involved like academic space , journalism , etc. Also , a lot of study on plagiarism has been announced but most of them lack the fundamental concept of optimizing tokens used for comparison between two documents. We , also studied thoroughly the graph based representation of the documents and demonstrated an approach to reduce the no. of suspected documents in the global corpus and also reduce the no. of tokens used to compare to detect plagiarism between two documents by customizing various optimization techniques.

In this work the standard optimization techniques have been studied improve the speed of comparison and sentence search in a document. In the proposed work , we improved the graph-based matching of documents by optimizing the no. of tokens involved and also improved the credibility of suspected documents searched in the global corpus for comparison to detect plagiarism .Both memory usage and computation time optimization was achieved.

This is especially useful where Internet connectivity is not stable or server platforms are not powerful to handle large no. of queries. This proposed work of ours will also be very beneficial in today’s world wide web world where no. of users is increasing day by day and in turn to meet the ever-increasing demands for decreased response time of the server.

Hence , for future we recommend these findings of ours to be used for querying and searching by search engines and also to reduce workload of plagiarism detection systems.

* 1. **Future Work**

We plan to incorporate all the available evolutionary optimization techniques to compare the results we have achieved. We also plan to use fuzzy rules to detect similarity of words despite spelling mistakes , separation tokens and difference or tenses and meanings.

This graph based matching technique and optimization in our proposed system could be incorporated in the traversal algorithms in the area of networking to increase speed to test for its usefulness in the field of telecommunications where each node here representing a concept could be replaced with nodes representing routers , IPAddresses , etc.or in mobile technologies to detect small changes in configurations or information stored in nodes where duplication is used as a basis of integrity test.

# References

[1] Engineering Optimization: An Introduction with Metaheuristic Applications. By Xin-

She Yang , John Wiley & Sons, Inc.(2010)

[2] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi. : The bees

algorithm - a novel tool for complex optimization problems, In Proceedings of IPROMS, July 2006, web conference , took place on internet , pp. 454-461.

[3] D. Karaboga, B. Basturk. : A powerful and Efficient Algorithm for Numerical Function

Optimization: Artificial Bee Colony (ABC) Algorithm, Journal of Global Optimization,volume 39(3), pp. 459-171, 2007.

[4] B. Basturk, D. Karaboga. : An Artificial Bee Colony (ABC) Algorithm for Numeric

function Optimization, In Proceedings of IEEE Swarm Intelligence Symposium,Indianapolis, Indiana, USA,pp. 654 - 658 , May 12-14, 2006.

[5] D. Karaboga, "An idea based on honey bee swarm for numerical optimization",Technical

Report TR06, Erciyes University, Turkey, October , 2005 , pp.1-10.

[6] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC)

algorithm", Applied Soft Computing, Volume 9, Issue 2, 2008 , pp. 687-697 .

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing",

Science,volume 220, No. 4598, 13 May ,1983 , pp. 671-680.

[8] Particle Swarm Optimization, James Kennedy and Russell Eberhart , paper appears in

Neural Networks, 1995. Proceedings., IEEE International Conference on, Nov/Dec

1995, volume 4 , pp.1942 – 1948.

[9] S. Nakrani and C. Tovey, "On honey bees and dynamic server allocation in Internet

hostbug centers", Adaptive Behavior*,* December 10 – 12 , 2004 , pp. 223-240.

[10] IEEE paper : A. Schenker, H. Bunke, M. Last, and A. Kandel, "Graph‐Theoretic

Techniques for Web Content Mining", Series in Machine Perception and Artificial

Intelligence, 62, World Scientific, 2005 , 2005 - 235

[11] IEEE paper : A.Schenkar, M Last, H. Bunke3, and A.Kandel,“Classification of web

documents using graph matching” , in International Journal of Pattern Recognition and Artificial Intelligence, Special Issue on Graph Matching in Computer Vision and Pattern Recognition, Vol. 18, No. 3, pp. 475-496, 2004.

[12] Journal of Computing : Plagiarism Detection Using Graph-Based Representation by

Ahmed Hamza Osman, Naomie Salim, Mohammed Salem Binwahlan , pp.36-41 , Journal of Computing , VOLUME 2, ISSUE 4, April , 2010.

[13] International Paper : KEA: Practical Automatic Keyphrase Extraction , Ian H. Witten,

Gordon W. Paynter,Eibe Frank,Carl Gutwin and Craig G. Nevill-Manning , 4th ACM Conference on Digital Libraries, Berkeley, CA, USA, August 11-14, 1999 , pp. 254-255

[14] EACL,99’ : Term Extraction + Term Clustering : An Integrated Platform for Computer

Aided Terminology , Didier Bourigault , Chritian Jaquemin , in 9th Conference of the European Chapter of the Association for Computational Linguistics , Bergen, Norway from 8 through 12 June, 1999 , pp. 15-22

[15] Gutwin, C., Paynter, G.W., Witten, I.H., Nevill- Manning, C.G. and Frank, E. (1998)

“Improving browsing in digital libraries with keyphrase indexes.” Technical Report,

Department of Computer Science, University of Saskatchewan , Published in:

Journal Decision Support Systems - From information retrieval to knowledge

management: enabling technologies and best practices , Canada , archive Volume 27

Issue 1-2, Nov. 1999 , pp.1-31

[16] Jones, S. (1998) “Link as you type : using key phrases for automated dynamic link

generation” Working Paper 98/16, Department of Computer Science, University of

Waikato, New Zealand , Sept. 1998 .

[17] Witten, I.H. (1999) “Browsing around a digital library.” *Proc. Australasian Computer*

*Science Conference*, Auckland, New Zealand, August 1993.pp. 1–14.

[18] R. Navigli and P. Velardi, “Learning Domain Ontologies from Document Warehouses

and Dedicated Web Sites," Computational Linguistics, vol. 30, pp. 151‐179,2004.

[19] I. Bichindaritz and S. Akkineni, “Concept Mining for Indexing Medical Literature,"

Lecture Notes in Computer Science, vol. 3587, 2005 , pp. 682‐692.

[20] Cicutto L. Plagiarism: avoiding the peril in scientific writing. Chest 2008 , October 25 –

30, 2008; Philadelphia, Pennsylvania , pp. 579–581.

[21] Green L. Reviewing the scourge of self-plagiarism. M/C Journal 2005;Volume-8(5).

<http://journal.media-culture.org.au/0510/07-green.php> .

[22] Rogers LF. , “Salami slicing, shotgunning, and the ethics of authorship” , AJR 1999 ,

Volume 173, Issue 4, October; pp.265.

[23] Chang-Keon Ryu,Hyong-Jun Kim,   Seung-Hyun Ji,   Gyun Woo and  Hwan-Gue Cho ,

“Detecting and tracing plagiarized documents by reconstruction plagiarism-evolution

tree , ” IEEE Int. Conf. Computer and Information Technology, Sydney,8-11 July 2008

(CIT 2008) , pp.119–124.

[24] Sergey Butakov and Vladislav Shcherbinin, “On the number of search queries required

for Internet plagiarism detection” in Proc. Of Ninth IEEE Int. Conf. Advanced Learning

Tech.,2009, Sydney, May 4-7 , 2009 , pp. 482-493.

[25] Heintze, N. , “ Scalable document fingerprinting” , Paper presented at the Second

USENIX Workshop on Electronic Commerce , Oakland, California ,November 18-21, 1996 , pp. 191–200.

[26] Lyon, C., Malcolm, J. A., & Dickerson, R. G. , “Detecting short passages of similar

text in large document collections” , Paper presented at the Conference on Empirical

Methods in Natural Language Processing, Carnegie Mellon University,

Pittsburgh, PA USA on June 3 – 4 , 2001 ,pp. 118–125.

[27] Yerra, R., & Ng, Y.‐K. , “A Sentence‐Based Copy Detection Approach for Web

Documents” , In Proceedings of 2005 International Conference on Fuzzy Systems and

Knowledge Discovery , Changsha, China, August 27-29,2005 , pp. 557‐570.

[28] Antonio, S., Hong Va, L., & Rynson, W. H. L. (1997).CHECK: a document plagiarism

detection system. Paper presented at the Proceedings of the 1997 ACM symposium on

applied computing , San Jose, CA, USA — February 28 - March 01, 1997 , pp.70-77.

[29] Manuel, Z., Marco, F., Massimo, M., & Alessandro, P. , “ Plagiarism Detection

through Multilevel Text Comparison” , Paper presented at the Second International

Conference on Automated Production of Cross Media Content for Multi‐Channel

Distribution , 13-15 Dec. 2006 , Leeds, UK , pp.181-185.

[30] S. Niezgoda and T. P.Way. Snitch: a software tool for detecting cut and paste plagiarism.

In SIGCSE ’06, Houston, Texas, USA March 1-5, 2006 ACM, 2006 , pp. 51–55.

[31] R. G. Paredes, J. A. Sanchez, and A. Razo. Drawing the line between fair use and

plagiarism for digital documents. In ENC 2007, 24-28 September 2007, Morelia,

Michoacan, Mexico, pp. 113–122.

[32] L. Prechelt, G. Malpohl, and M. Philippsen. Finding plagiarisms among a set of

programs with JPlag. Journal of Universal Computer Science, 8(11) , 2002 , pp.1016–

1038.

[33] A. Si, H. V. Leong, and R. W. H. Lau. Check: a document plagiarism detection system.

In SAC ’97, ACM, 1997 ,February , 28th – March ,2nd , 1997 , San Jose, CA , pp. 70–77.

[34] M. J. Wise. , “Detection of similarities in student programs: Yap’ing may be preferable

to plague’ing” . ACM SIGSCE Bulletin (Proc. of 23rd SIGCSE Technical Symp), 24(1):

ACM, March ,1999 , pp.268–271.

[35] A. Aiken. Moss (measure of software similarity) plagiarism detection system. 1998.

[36] D. Gitchell and N. Tran. Sim: a utility for detecting similarity in computer programs. In

SIGCSE ’99 , New Orleans, LA, USA — March 24 - 28, 1999, pp. 266–270.

[37] S. Gruner and S. Naven. Tool support for plagiarism detection in text documents. In 20th

Annual ACM Symposium on Applied Computing Santa Fe, New Mexico, March 13 -17,

pp. 776–781.

[38] J.-H. Ji,G.Woo and H.-G. Cho.A source code linearization technique for detecting

plagiarized programs.In June 25–27, 2007,Dundee, Scotland,United Kingdom,pp.73–77.

[39] SHEN Yang , LI Shu-chen, TIAN Chen-geng and CHENG Ming , “Research on Anti-

plagiarism System and the Law of Plagiarism” , in First International Workshop on

Education Technology and Computer Science , 7-8 March,2009 , Wuhan, Hubei, China ,

pp.296 **–** 300

[40] CopyCatch Software Suite. Accessed June , 20 , 2011 at http://www.copycatchgold.com/

[41] EVE2 Plagiarism detection for teachers. Accessed June , 28 , 2011 at http://www.canexus.com/

[42] MyDropBox Internet Plagiarism Detection Service . Accessed June , 28 , 2011 at http://www.mydropbox.com.

[43] The PAIRWise system. Accessed June , 28 , 2011 at http://www.pairwise.cits.ucsb.edu/system.htm

[44] Schleimer S., Wilkerson D., and Aiken A. (2003).Winnowing: Local Algorithms for

Document Fingerprinting.Proceedings of the ACM SIGMOD International Conference

on Management of Data, June 2003 ,pp. 76-85.

[45] Turnitin: The WriteCycle Suite. Brochure. Accessed November, 10, 2008 at

<http://turnitin.com/static/products.html>

[46] Chang-Keon Ryu,Hyong-Jun Kim,   Seung-Hyun Ji,   Gyun Woo and  Hwan-Gue Cho ,  “Detecting and tracing plagiarized documents by reconstruction plagiarism-evolution tree , ”

IEEE Int. Conf. Computer and Information Technology,January 3 - 5th, 2008 ,Salt Lake

City Uttah , pp. 119 - 124 .

[47] Lifang Han, Baojiang Cui,,Ru Zhang, Zhongxian Li , Jianxin Wang and Yongle Hao**,**

**“**Type Redefinition Plagiarism Detection of Token-Based Comparison” , in Int Conf.

on Multimedia Information Networking and Security MINES, Nanjing, China , 4-6

Nov., 2010 , pp.351-355.

[48] History of optimization, <http://hse-econ.fi/kitti/opthist.html>

[49] The Turing Archive for the History of Computing, <http://www.alanturing.net/>

[50] Ontology Learning : <http://en.wikipedia.org/wiki/Ontology_learning>

[51] Plagiarism : http://en.wikipedia.org/wiki/Plagiarism