A Project Report

<PLAGIARISM CHECK USING

DYNAMIC PROGRAMMING>

# Bachelor of Technology In

Design and Analysis of Algorithms

By

NAVADEEP REDDY – 2010030313

SIDDARTH – 2010030475

VIPUL REDDY – 2010030502

MANOJ PERAVALI – 2010030502

Under the supervision of

# Dr. UDAYA RANI GURALA

# Assistant Professor



Department of Computer Science and Engineering

K L University Hyderabad,

Aziz Nagar, Moinabad Road, Hyderabad – 500 075, Telangana, India.

March, 2022

**Declaration**

## The Project Report entitled “<PLAGARISM CHECK USING DYNAMIC PROGRAMMING

>”is a record of Bonafede work of < NAVADEEP – 2010030313, SIDDARTH – 2010030475, VIPUL REDDY – 2010030502, MANOJ PERAVALI – 2010030503>, submitted in partial fulfillment for the award of B. Tech in the Department of Computer Science and Engineering to the K L University, Hyderabad. The results embodied in this report have not been copied from any other Departments/University/Institute.

<Signature of the Students >

**Certificate**

This is to certify that the Social Internship Report entitled “NETWORK SIMULATION USIMG CISCO PACKET TRACER” is being submitted by NAVADEEP REDDY–2010030313, SIDDARTH – 2010030475, VIPUL REDDY – 2010030502, MANOJ PERAVALI – 2010030503 submitted in partial fulfillment for the award of B.Tech in Computer Science and technology(C.S.E) to the K L University, Hyderabad is a record of Bonafede work carried out under our guidance and supervision.

The results embodied in this report have not been copied from any other departments/ University/Institute.

## Signature of the Supervisor

Dr UDAYA RANI GURALA

## Signature of the HOD Signature of the External Examiner

**ACKNOWLEDGEMENT**

First and foremost, we thank the lord almighty for all his grace & mercy showered upon us, for completing this Social Internship successfully.

We take grateful opportunity to thank our beloved Founder and Chairman who has given constant encouragement during our course and motivated us to do this Social Internship. We are grateful to our Principal **Dr. L. Koteswara Rao** who has been constantly bearing the torch for all the curricular activities undertaken by us.

We pay our grateful acknowledgement & sincere thanks to our Head of the Department **Dr. Chiranjeevi Manike** for his exemplary guidance, monitoring and constant encouragement throughout the course. We thank Dr UDAYA RANI GURALAof our department who has supported throughout this course holding a position of supervisor.

We whole heartedly thank all the teaching and non-teaching staff of our department without whom we won’t have made this Social Internship a reality. We would like to extend our sincere thanks especially to our parent, our family members and friends who have supported us to make this Social Internship a grand success.

***ABSTRACT***

* We all know that **computers** can only understand *0s* and *1s*, and for us to perform some **computation** on textual data we need a way to **convert** the *text* into *numbers*.
* **Word embedding**
* The process of converting the *textual data* into an array of numbers is generally known as **word embedding**.
* The vectorization of textual data to vectors is not a **random** process instead it follows certain **algorithms** resulting in words being represented as a **position** in space. we going to use **scikit-learn** built-in features to do this.

**TABLE OF CONTENTS**

* 1. Introduction………………………………………………vii
  2. Literature Survey…………………………………………
  3. Hardware & Software requirements……………………...viii
  4. Implementation…………………………………………...ix
  5. Results ……………………………………………………x
  6. References………………………………………………...xi

**INTRODUCTION:**

Plagiarism is alluded to as replicating another’s work or obtaining another person’s unique thoughts. Numerous understudies are blamed for counterfeiting because they basically don’t comprehend it and, along these lines, don’t realize how to keep away from it. When another author thought, expressions, original work and thoughts are stolen and published by another person is wrong and is regarded as plagiarism. Additionally, when words of another are stolen and passed off as one original work, failure to cite someone else work and present them as original and new ideas without crediting the source committing literary theft that’s plagiarism. Plagiarism is an expression of extortion where someone else work is stolen and afterwards it is lied about its source. United states law states that original ideas expression is regarded as intellectual property and laws of copyrights protect it as it is done to inventions that are original. If expression are recorded in books or computer files, they are under copyrights protection.

**LITERACTURE REVIEW**

Similarity Detection in Java Programming Assignments-----Mohamed El Bachir Menai, Nailah Salah Al----Hassoun-August 24–27, 2018

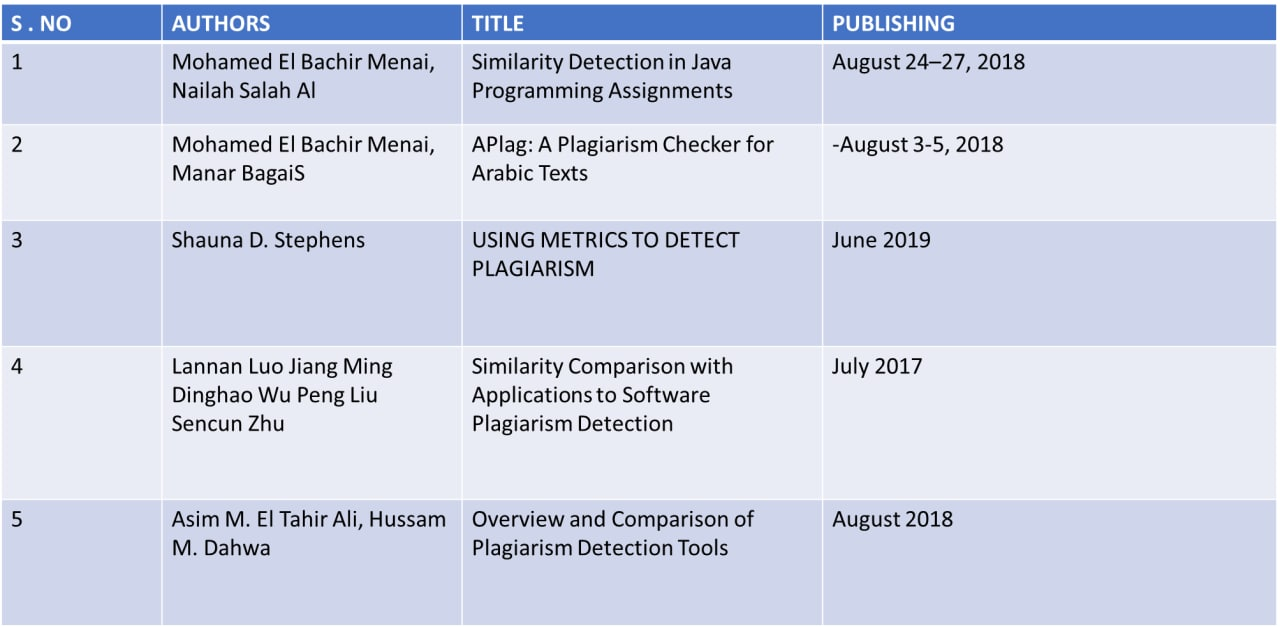
APlag: A Plagiarism Checker for Arabic Texts----Mohamed El Bachir Menai, Manar BagaiS-August 3-5, 2018

USING METRICS TO DETECT PLAGIARISM---Shauna D. Stephens---June 2019

Similarity Comparison with Applications to Software Plagiarism Detection----Lannan Luo Jiang Ming Dinghao Wu Peng Liu Sencun Zhu

The latest developments in plagiarism detection in medical literatur---Thorakkal Shamim----july 2017

Overview and Comparison of Plagiarism Detection Tools-------Asim M. El Tahir Ali, Hussam M. Dahwa Abdulla, and V´aclav Sn´aˇsel-------August 2018



**SOFTWARE AND HARDWARE REQUIREMENTS:**

OS: Windows RAM: 8.00 GB

Processor: Intel Core 5

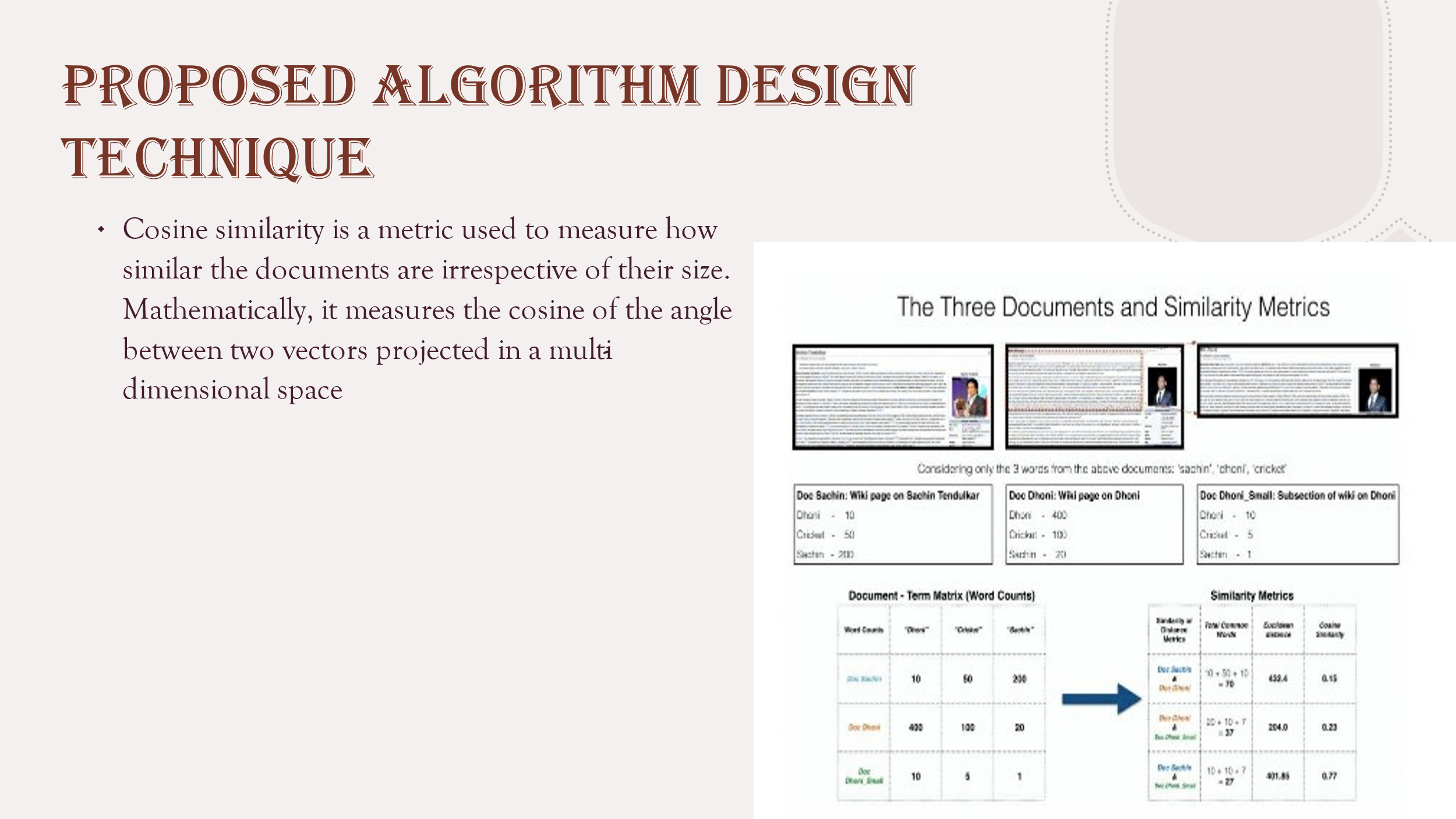
System-type: 64-bit OS

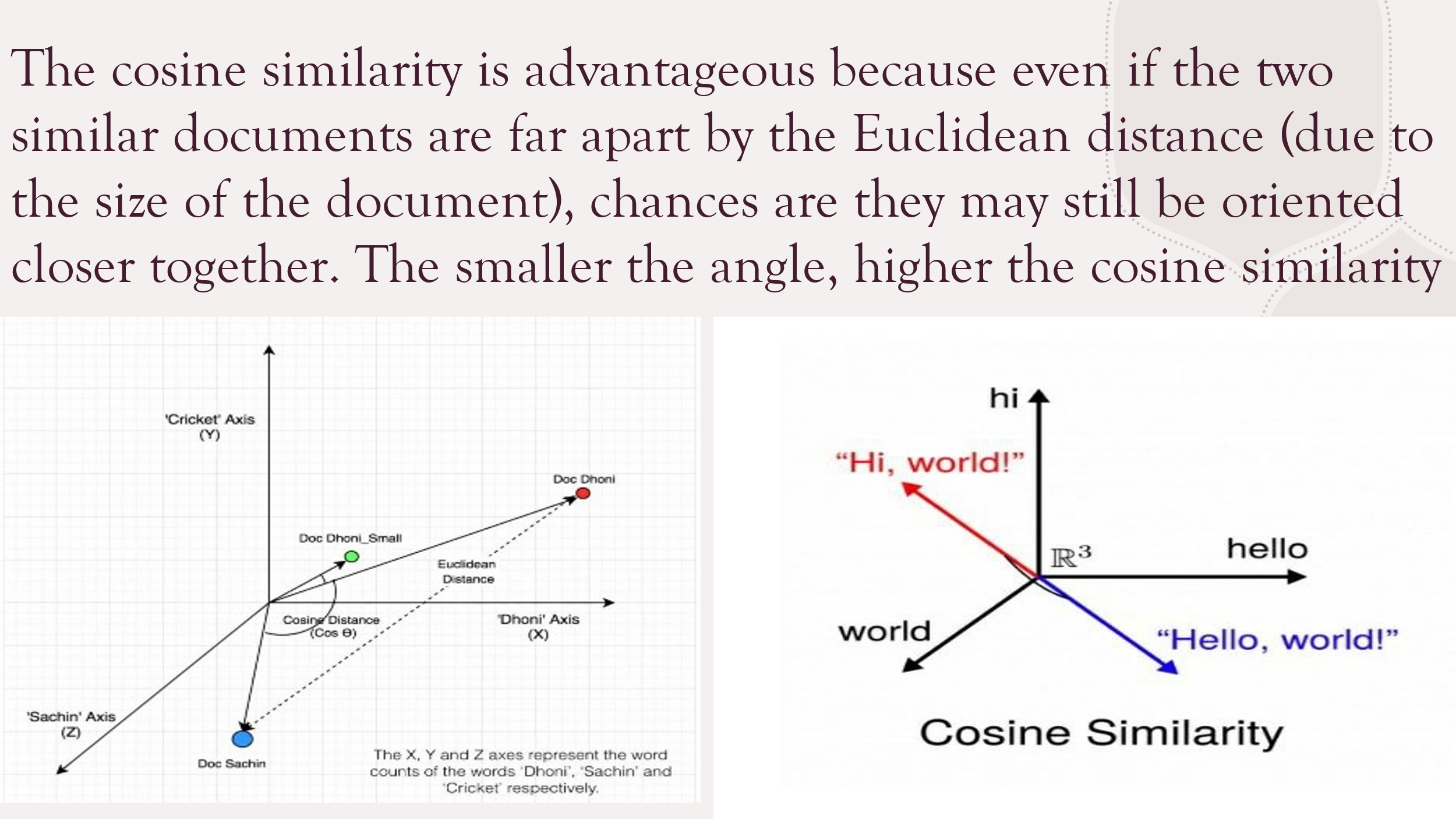
Version: 20H2

Edition: Windows 10

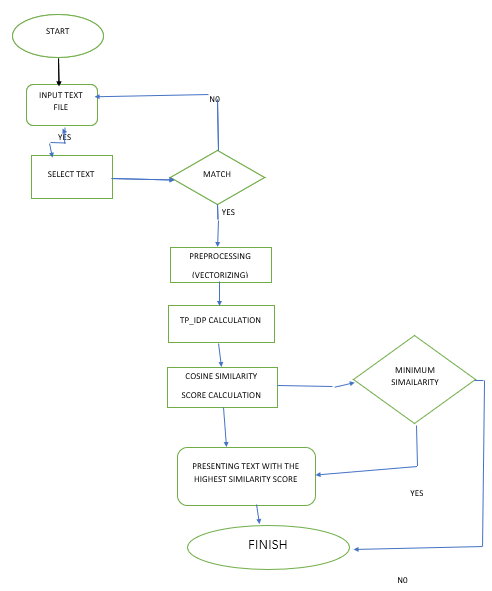
Tools: PyCharm, GitHub

**ALGORITHMS**





**FLOW CHART**



**CODE**

import os

from numpy import vectorize

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

sample\_files = [doc for doc in os.listdir() if doc.endswith('.txt')]

sample\_contents = [open(File).read() for File in sample\_files]

vectorize = lambda Text: TfidfVectorizer().fit\_transform(Text).toarray()

similarity = lambda doc1, doc2: cosine\_similarity([doc1, doc2])

vectors = vectorize(sample\_contents)

s\_vectors = list(zip(sample\_files, vectors))

def check\_plagiarism():

results = set()

global s\_vectors

for sample\_a, text\_vector\_a in s\_vectors:

new\_vectors = s\_vectors.copy()

current\_index = new\_vectors.index((sample\_a, text\_vector\_a))

del new\_vectors[current\_index]

for sample\_b, text\_vector\_b in new\_vectors:

sim\_score = similarity(text\_vector\_a, text\_vector\_b)[0][1]

sample\_pair = sorted((sample\_a, sample\_b))

score = sample\_pair[0], sample\_pair[1], sim\_score

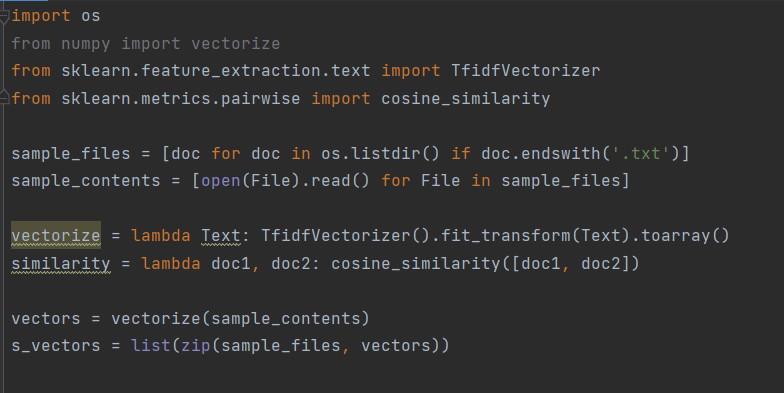
results.add(score)

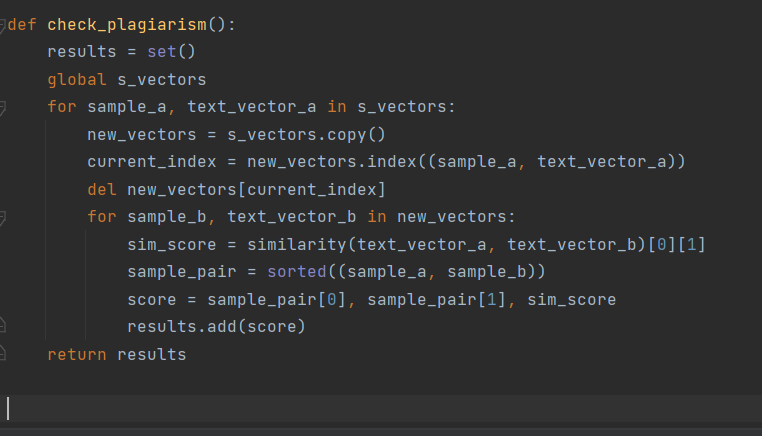
return results

for data in check\_plagiarism():

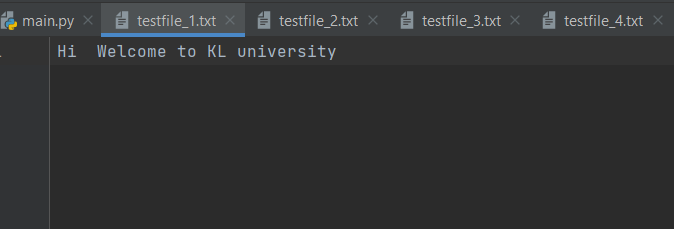
print(data)

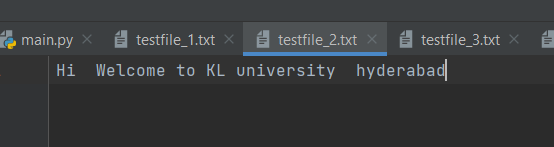
**IMPLEMENTATION**

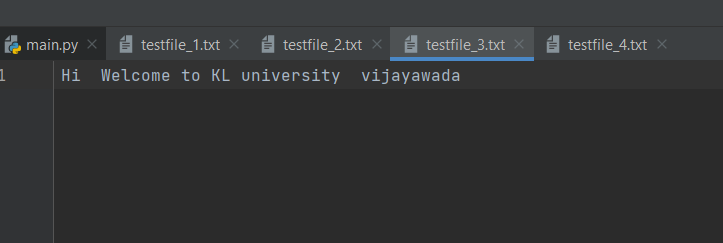


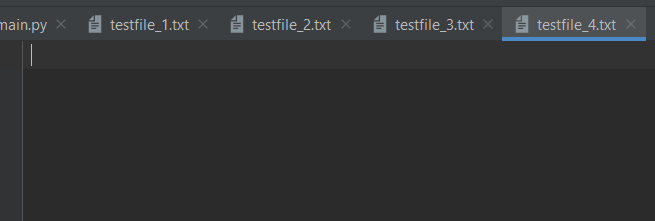


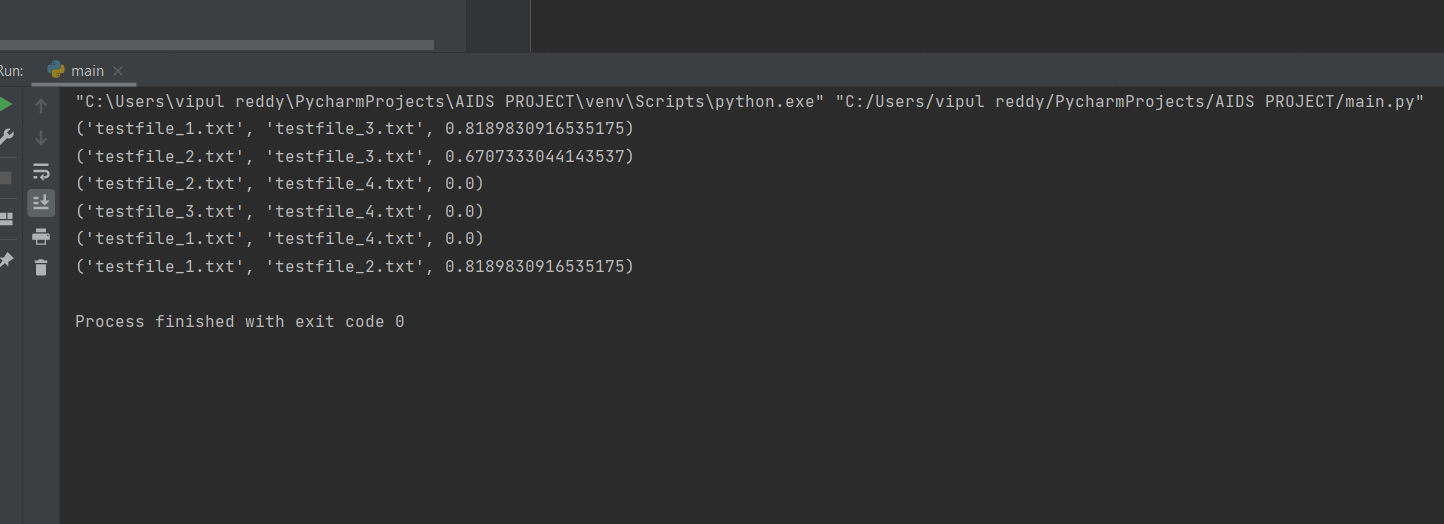
**OUTPUTS**











**FUTURE SCOPES OF PROJECT & CONCLUSION**

The result is students should be a better academic writer if they want to avoid plagiarism. If those English-speaking countries can solve the above problems and enable students to understand the value of doing assignments, less plagiarism will appear in the future.

I’d like to examine three sectors that could benefit from a similar plagiarism checker product – art, music, and video.  All of these industries currently have a number of outstanding legal cases and disputes over plagiarized content.   With the right technology, plagiarism in all of these fields could be minimized through detection and checking prior to distribution, we aim to conduct more experiments on larger texts to detect the text re-use in long documents such as theses and dissertations with a reasonable computational complexity

**REFERENCES**

[1] ACL. http://savannah.nongnu.org/projects/acl, 2013.

[2] AdvanceCOMP. http://advancemame.sourceforge.net/compdownload.html.

[3] Attr. http://savannah.nongnu.org/projects/attr, 2013.

[4] B. S. Baker. On finding duplication and near-duplication in large software systems. In WCRE’95, pages 86–95, 1995.

[5] G. Balakrishnan and T. Reps. WYSINWYX: What You See Is Not What You eXecute. ACM Transactions on Programming Languages and Systems (TOPLAS), 32(6):23:1–23:84, Aug. 2010.

[6] G. Balakrishnan, T. Reps, D. Melski, and T. Teitelbaum. WYSINWYX: What You See Is Not What You eXecute. In Verified Software: Theories, Tools, Experiments (VSTTE), pages 202–213,

2005.

[7] BAP: The Next-Generation Binary Analysis Platform. http://bap.ece.cmu.edu/, 2013.

[8] Binary Diff (bdiff). http://sourceforge.net/projects/bdiff/, 2013.

[9] Bzip2. http://bzip.org/index.html, 2010.

[10] C. Cadar, D. Dunbar, and D. Engler. KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs. In OSDI’08, 2008.

‘[11] C. Cadar and D. Engler. Execution generated test cases: How to make systems code crash itself. In SPIN’05, 2005.

[12] C. Cadar, V. Ganesh, P. Pawlowski, D. Dill, and D. Engler. EXE: Automatically generating inputs of death. In CCS’06, 2006.

[13] CLOC—Count Lines of Code. http://cloc.sourceforge.net/, 2013.

[14] C. Collberg, C. Thomborson, and D. Low. A taxonomy of obfuscating transformations. Technical Report 148, The Univ. Auckland, 1997.

[15] Compuware-IBM Lawsuit. http://news.zdnet.com/2100-3513 22-5629876.html, 2013.