ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

Mümine KAYA

PLAGIARISM DETECTION SUPPORTED BY WEB USAGE MINING FOR TURKISH TEXTUAL DOCUMENTS

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ADANA, 2015

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PLAGIARISM DETECTION SUPPORTED BY WEB USAGE MINING FOR TURKISH TEXTUAL DOCUMENTS

Mümine KAYA

PhD THESIS

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the Doctor of Philosophy by the board of jury on 16/02/2015.

…….....	……..….	..…..…..…..…..…..……
Assoc. Prof. Dr. S. Ayşe ÖZEL	Assoc. Prof. Dr. Ulus ÇEVİK	Assoc. Prof. Dr. Mutlu AVCI
SUPERVISOR	MEMBER	MEMBER

……	…..…..…..…..…..…..…..…..….
Prof. Dr. Hamza EROL	Assoc. Prof. Dr. Yücel SAYGIN
MEMBER	MEMBER

This PhD Thesis is written at the Department of Institute of Natural And Applied Sciences of Çukurova University.
Registration Number:

Prof. Dr. Mustafa GÖK
Director
Institute of Natural and Applied Sciences

This thesis was supported by the TUBITAK 2211-C the Priority Areas Related to Domestic Doctoral Scholarship and Scientific Research Project Unit of Çukurova University for my thesis (Project Number: MMF2013D8).

Note: The usage of the presented specific declerations, tables, figures, and photographs either in this thesis or in any other reference without citiation is subject to "The law of Arts and Intellectual Products" number of 5846 of Turkish Republic

ABSTRACT

PhD THESIS

PLAGIARISM DETECTION SUPPORTED BY WEB USAGE MINING FOR TURKISH TEXTUAL DOCUMENTS

Mümine KAYA

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Supervisor : Assoc. Prof. Dr. Selma Ayşe ÖZEL
Year: 2015, Pages: 220
Jury : Assoc. Prof. Dr. Selma Ayşe ÖZEL
: Assoc. Prof. Dr. Mutlu AVCI
: Assoc. Prof. Dr. Ulus ÇEVİK
: Prof. Dr. Hamza EROL
: Assoc. Prof. Dr. Yücel SAYGIN

The aim of this thesis is to compare the similarities among textual documents in Turkish, specifically student assignments from Çukurova University Department of Computer Engineering Moodle web site, and to detect plagiarism with a string matching algorithm. Since existing systems are developed for English, and there are not many systems for Turkish, we designed this study for Turkish language. We provide detection of similarities when a change was done in the annexes of the words, the synonyms of words were used, and phrase, sentence or paragraph were transposed. We also aim to improve the results of the similarity detection by analyzing the behavior of the students on Moodle using web usage mining methods.

This study is focused on the solutions for detecting plagiarism which has increased due to the Internet usage in the academic environment. In the proposed system student assignments were compared with each other using the Greedy String Tiling Algorithm and Cosine Similarity methods. We observed that using the plagiarism detection tool reduces time and effort needed for the assessment of the textual assignments, prevents our students from plagiarizing, and increases their success in their course. The proposed plagiarism detection tool was compared with the existing tools and methods, and its success was evaluated over sample texts.

Key Words: Plagiarism, Turkish Documents, Web Usage Mining, String Matching Algorithms and Distance Metrics, Moodle Distance Education System

ÖZ

DOKTORA TEZİ

TÜRKÇE METİNSEL DÖKÜMANLAR İÇİN WEB KULLANIM MADENCİLİĞİ İLE DESTEKLENMİŞ İNTİHAL TESPİTİ

Mümine KAYA

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Doç. Dr. Selma Ayşe ÖZEL
Yıl: 2015, Sayfa: 220
Jüri : Doç. Dr. Selma Ayşe ÖZEL
: Doç. Dr. Mutlu AVCI
: Doç. Dr. Ulus ÇEVİK
: Prof. Dr. Hamza EROL
: Doç. Dr. Yücel SAYGIN

Bu çalışmanın amacı; Çukurova Üniversitesi Bilgisayar Mühendisliği Bölümü için hazırlanan Moodle tabanlı Uzaktan Eğitim Sitesine yüklenen Türkçe ile yazılmış metinsel dokümanlar arasındaki benzerlikleri karşılaştırmak ve intihalleri bir metin eşleme algoritması ile tespit etmektir. Var olan sistemlerin genellikle İngilizce dili için geliştirilmesinden ve Türkçe için kullanılan yeterli sayıda sistem bulunmamasından dolayı bu çalışmayı Türkçe dili için gerçekleştirmekteyiz. Kelime eklerinde değişiklik olduğunda, kelimelerin eşanlamlısı kullanıldığında, kelime grupları, cümle ya da paragrafların yerleri değiştirildiğinde de benzerliklerin tespitini sağlamaktayız. Moodle'daki öğrencilerin davranışlarını da web kullanım madenciliği yöntemleri kullanılarak analiz ederek benzerlik tespitinin sonuçlarını geliştirmeyi amaçlamaktayız.

Bu çalışmada, İnternetin akademik ortamda çok kullanılmasıyla artan intihal probleminin çözümleri üzerinde durulmaktadır. Önerilen yöntemde öğrenci ödevlerinin birbirleriyle benzerlikleri Hırslı Metin Eşleme Algoritması ve Kosinüs Benzerliği yöntemleri kullanılarak karşılaştırılmıştır. İntihal tespiti algoritmalarının kullanılmasının metinsel ödevlerin değerlendirilmesi için gereken zamanı ve eforu azalttığı, öğrencilerin intihal yapmasını önlediği ve dersteki başarılarının arttığı gözlenmiştir. Önerilen intihal tespit aracı mevcut araç ve yöntemlerle karşılaştırılmış, örnek metinler üzerinde başarısı sınanmıştır.

Anahtar Kelimeler: İntihal, Türkçe Dokümanlar, Web Kullanım Madenciliği, Metin Eşleme Algoritmaları ve Uzaklık Ölçütleri, Moodle Uzaktan Eğitim Sistemi

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Assoc. Prof. Dr. Selma Ayşe ÖZEL, for her supervision guidance, encouragements, patience, and motivation, constructive and lead-in and useful suggestions and her valuable time for this work. I am grateful to her for all the things that I have learned under her supervision. This dissertation would not have been possible without the help of my advisor, who I have enjoyed studying with her. She never deprived her help at any stage of my work. I deeply thank my eternal gratitude to her to support me for selecting the thesis about "Plagiarism Detection Supported by Web Usage Mining for Turkish Textual Documents", to share her knowledge, suggestions, experience and opinion during all the time of my studies, to give me some guidance for my thesis. I will always take my estimable advisor, Assoc. Prof. Dr. Selma Ayşe ÖZEL as my model during my career.

I would like to show gratitude to members of my PhD thesis jury, Assoc. Prof. Dr. Mutlu AVCI, for his experience and suggestions, his lead-in and affirmative contribution at every stage of my work, Assoc. Prof. Dr. Ulus ÇEVİK, for his constructive and lead-in suggestions and corrections, Prof. Dr. Hamza EROL and Assoc. Prof. Dr. Yücel SAYGIN, for their constructive suggestions and corrections. I would also like to thank Prof. Dr. Mehmet TÜMAY for his experience and lead-in suggestions, and Asst. Prof. Dr. Alper Kamil DEMİR for his constructive suggestions.

I gratefully thank the Faculty members of Çukurova University Computer Engineering Department, and Adana Science and Technology University sincerely. Additionally, I especially like to show my gratitude to my friends Res. Asst. Ayşegül UYSAL GLINIECKI, Res. Asst. Hediye KIRLI AKIN, and Res. Asst. Abdullah Emre KELEŞ, who supported me during my studies. I owe them for their valuable help and encouragement.

I profoundly express appreciation to TUBITAK 2211-C the Priority Areas Related to Domestic Doctoral Scholarship (Application Number: 1649B031402174) for their support.

I genuinely acknowledge Scientific Research Project Unit of Çukurova University - under the Grant No MMF2013D8 - for their support.

Last but not the least important, I would like to thank my family to stand by me since the beginning of my education life and my career, to vitalize me with their presence, to support me with a huge patience always, not to deprive their physical and moral help from me. None of this could have happened without my family. I heartedly say thank you to my dear father, Lawyer İsmet KAYA, and my dear mother, Cavidan KAYA, to be my lifeblood with their presence, my sweetie sister, Agricultural Engineer CPD Emine KAYA, for her help, support and encouragements on my studies.

CONTENTS **PAGE**

ABSTRACT I
ÖZ II
ACKNOWLEDGEMENTS III
CONTENTS V
LIST OF TABLES IX
LIST OF FIGURES XIII
LIST OF ABBREVIATONS XV
l. INTRODUCTION 1
2. PREVIOUS WORK 7
2.1. Distance Education 7
2.1.1. Distance Education Tools 9
2.1.2. Comparison of Blackboard & Moodle 14
2.2. Plagiarism 15
2.2.1. Types of Plagiarism 17
2.2.2. Automated Plagiarism Detection 20
2.2.2.1. Plagiarism Detection in Natural Language Texts 20
2.2.2.2. Plagiarism Detection in Turkish Texts 23
2.2.2.3. Plagiarism Detection in Source Codes 25
2.2.2.4. Tools for Plagiarism Detection 27
2.2.2.5. Plugins and Tools for Moodle 38
2.2.2.6. Algorithms for Plagiarism Detection 41
2.3. String Matching Algorithms 44
2.3.1. Exact String Matching Algorithms 44
2.3.1.1. Karp-Rabin Algorithm 45
2.3.1.2. Morris Pratt Algorithm 45
2.3.1.3. Boyer-Moore Algorithm 46
2.3.1.4. Quick Search Algorithm 46
2.3.1.5. Zhu-Takaoka Algorithm 47
2.3.1.6. Berry-Ravindran Algorithm 47

2.3.1.7. Optimal Mismatch Algorithm 47
2.3.1.8. Skip Search Algorithm 48
2.3.1.9. Knuth-Morris-Pratt Algorithm 48
2.3.2. Approximate String Matching Algorithms 48
2.3.2.1. Levenshtein Distance 49
2.3.2.2. Needleman-Wunsch 51
2.3.2.3. Smith-Waterman 52
2.3.2.4. Smith-Waterman Gotoh 53
2.3.2.5. Smith-Waterman Gotoh Affine Window 53
2.3.2.6. Hamming Distance 53
2.3.2.7. Jaro – Winkler 54
2.3.2.8. Q-Grams 55
2.3.2.9. The Winnowing Algorithm 55
2.3.2.10. Block Distance 56
2.3.2.11. Cosine Similarity 57
2.3.2.12. Euclidean Distance 57
2.3.2.13. Monge-Elkan 58
2.3.2.14. Jaccard Distance 58
2.3.2.15. Overlap Coefficient 59
2.3.2.16. Dice Coefficient 59
2.3.2.17. Minkowski Distance 60
2.3.2.18. Greedy String Tiling Algorithm 60
2.3.2.19. Heckel's Algorithm 61
2.3.2.20. Running Karp-Rabin Greedy String Tiling Algorithm 61
2.3.2.21. Longest Common Subsequence 62
2.4. Stemming and Thesaurus 62
2.5. Web Mining and Data Mining for LMS 66
3. MATERIAL AND METHOD 77
3.1. Material 77
3.1.1. Moodle 77
3.1.2. Turkish Dataset 77

3.1.3. Stemming 78
3.1.4. The Dictionary of Synonyms 78
3.1.5. The Abbreviation List 81
3.1.6. The Glossary of Computer Terms 81
3.1.7. The Turkish Stopword List 82
3.1.8. The Phrase List 84
3.1.9. String Matching Algorithms To Be Used 85
3.1.9.1. Cosine Similarity 87
3.1.9.2. Greedy String Tiling Algorithm 87
3.2. Method 89
3.2.1. Pre-processing Phase 91
3.2.2. Similarity Phase 92
3.2.2.1. GST Based Text Line Matching Algorithm 93
3.2.2.2. GST Based Punctuation Matching Algorithm 93
3.2.2.3. GST Based Stopword Matching Algorithm 94
3.2.2.4. Computation of Similarity Score 94
3.2.2.5. Integration of the Proposed System into Moodle Environment 97
3.2.3. Web Usage Mining on Moodle Logs 101
4. RESULTS AND DISCUSSION 105
4.1. Experiments for Determining the Need of Plagiarism Detection Tool Usage 105
4.2. Experiments for Determining the Distance Metrics 112
4.3. Experiments for Determining the Distance Metrics After The Preprocessing Phases 118
4.4. Experiment for Comparing Cosine Similarity with Other Well-Known Metrics 119
4.5. A Comparison of Greedy String Tiling Algorithm with Cosine Similarity 120
4.6. Experiments with Greedy String Tiling Algorithm 122
4.7. Effect of Using Dictionaries 122

4.8. Comparison of the Proposed System with the Existing Tools 123
4.9. Experiments with Student's Homeworks 136
4.10. The Comparison of the Running Time of All Similarity Tools 143
4.11. Using Data Mining for Determining the Effect of the User Behavior in Similarity Detection 143
4.12. Using Data Mining for Determining the Effect of the User Behavior in Grade Prediction 146
4.13. The Comparison of the Proposal with the Related Studies 149
5. CONCLUSION 151
REFERENCES 153
CURRICULUM VITAE 179
APPENDICES 180

LIST OF TABLES **PAGE**

Table 2.1. The distance education tools and web addresses 10
Table 2.2. Comparison of Distance Education Tools 15
Table 2.3. Free Plagiarism Detection Tools..... 27
Table 2.4. Commercial Plagiarism Detection Tools 28
Table 2.5. The comparison between the free textual plagiarism detection tools and web systems..... 182
Table 2.6. The comparison between the commercial textual plagiarism detection tools and web systems 186
Table 3.1. A Small Portion of Our Synonyms Dictionary 80
Table 3.2. A Small Portion of Our Stemmed Synonyms Dictionary 80
Table 3.3. A Small Portion of Our Abbreviation List..... 81
Table 3.4. A Small Portion of our Computer Science Term List 82
Table 3.5. Turkish Stopword List 83
Table 3.6. A Small Portion of Preprocessed Turkish Stopword List..... 84
Table 3.7. An Example of Our Turkish and English Computer Science Phrase List..... 85
Table 3.8. The Pseudocode of Greedy String Tiling Algorithm..... 89
Table 3.9. The Preprocessing Steps for a Sentence in Our System 92
Table 3.10. The 2x2 Confusion Matrix In Classification 103
Table 3.11. A Confusion Matrix for One More than Class 104
Table 4.1. Students' Plagiarism Behavior for Assignments in 2010 Fall Semester..... 106
Table 4.2. Students' Plagiarism Behavior for Assignments in 2011 Fall Semester..... 107
Table 4.3. Students' Performances for the Data Structures Course in 2010 and 2011 Fall Semesters 108
Table 4.4. Students' Plagiarism Behavior for Assignments in 2012 Fall Semester..... 109

Table 4.5. Students' Plagiarism Behavior for Assignments in 2013 Fall Semester 111
Table 4.6. Students' Performances for the Data Structures Course in 2012 and 2013 Fall Semesters 112
Table 4.7. The results of the first experiment with the distance metrics 114
Table 4.8. The results of the second experiment with the distance metrics..... 115
Table 4.9. The results of the third experiment with the distance metrics 116
Table 4.10. Three Texts Created From The YTÜ Kemik Web Site 190
Table 4.11. The results of the experiments of Section 4.2 117
Table 4.12. The Experimental Results for Determining the Distance Metrics After The Preprocessing Phases 190
Table 4.13. Two Texts Created From The Homeworks 191
Table 4.14. The Preprocessing Methods 119
Table 4.15. The Similarity Values for Each Distance Metric 120
Table 4.16. The compared datasets 121
Table 4.17. Similarity Scores Computed by GST and Cosine Similarity 121
Table 4.18. The used assignments in the experiment of Section 4.5 192
Table 4.19. Similarity Scores Computed by GST and Cosine Similarity for homeworks 121
Table 4.20. The results and the comparisons of the preprocessing for GST 122
Table 4.21. The compared datasets for preprocessing 123
Table 4.22. The comparisons of GST (m=1), GST (m=2), and GST(m=1 & no preprocess) 123
Table 4.23. Text1 and Text2 124
Table 4.24. Text3 and Text4 125
Table 4.25. The comparison result of the first dataset (% Similarity) 126
Table 4.26. The comparison result of the second dataset (% Similarity) 127
Table 4.27. The comparison result of the third dataset (% Similarity) 128
Table 4.28. The comparison result of the fourth dataset (% Similarity) 129
Table 4.29. The comparison result of the fifth dataset (% Similarity) 130
Table 4.30. The comparison result of the sixth dataset (% Similarity) 131

Table 4.31. The comparison result of the seventh dataset (% Similarity) 132
Table 4.32. The comparison result of the eighth dataset (% Similarity) 133
Table 4.33. The comparison result of the texts in 4.5 (% Similarity)..... 135
Table 4.34. The comparison result of the texts in Section 4.7 (% Similarity)..... 136
Table 4.35. The used assignments in the experiment of Section 4.9 198
Table 4.36. The Similarity Results of JPlag with m=2 token 204
Table 4.37. The Similarity Results of Ferret with m=3 token 204
Table 4.38. The Similarity Results of CopyCatch Gold with m=5 character 204
Table 4.39. The Similarity Results of WCopyfind with m=2 token 204
Table 4.40. The Similarity Results of SIM with m=2 token..... 205
Table 4.41. The Similarity Results of Sherlock with m=2 token..... 205
Table 4.42. The Similarity Results of Text_Matching 205
Table 4.43. The Similarity Results of Document Similarity–Cosine with m=2 token..... 205
Table 4.44. The Similarity Results of Document Similarity–Jaccard with m=2 token..... 206
Table 4.45. The Similarity Results of Document Similarity–Dice with m=2 token..... 206
Table 4.46. The Similarity Results of Document Similarity-Tanimoto with m=2 token 206
Table 4.47. The Results of our implemented Cosine Similarity..... 207
Table 4.48. The Results of our implemented Dice Similarity 207
Table 4.49. The Results of our implemented Jaccard Similarity..... 207
Table 4.50. The Similarity Results of DOCCop with m=8 token..... 207
Table 4.51. The Results of Turnitin 208
Table 4.52. The Proposed Algorithm Results As a Table for Scenario 28 with m=1 208
Table 4.53. The Proposed Algorithm Results As a Table for Scenario 28 with m=2 208
Table 4.54. The Proposed Algorithm Results of Stopwords as a Table with m=1 208

Table 4.55. The Proposed Algorithm Results of Punctuations as a Table with m=1 209
Table 4.56. The Results of the Comparisons 142
Table 4.57. The Comparison of the Running Time of All Similarity Tools..... 210
Table 4.58. The comparison of Apriori tests 144
Table 4.59. The most important Apriori Rules for the datasets..... 145
Table 4.60. The F-Measure Comparison of All Classification Algorithms for the raw dataset with all attributes 213
Table 4.61. The comparison of all experiments for plagiarism with 10-fold cross validation 146
Table 4.62. The comparison of Apriori tests 147
Table 4.63. The most important Apriori Rules for the datasets..... 147
Table 4.64. The comparison of all experiments in web usage mining for grade... 149
Table 4.65. Comparison of Our Proposal With the Related Studies The Comparison of Literature Review..... 214
Table 4.66. Comparison of study with the related studies about the web usage mining 217

LIST OF FIGURES **PAGE**

Figure 3.1. Main Architecture of the Proposed System 90
Figure 3.2. The Proposed String Matching Algorithm..... 96
Figure 3.3. Moodle Web Page of Submitted Assignment..... 97
Figure 3.4. Moodle Detailed Web Page of Submitted Assignment 98
Figure 3.5. The Result Web Page of the Proposed Algorithm..... 99
Figure 3.6. The Detailed Similarity Report..... 100
Figure 3.7. Numbers of Matched Tokens..... 100
Figure 3.8. Processing Time of the Algorithms 101

ABBREVIATONS

AOSO	: Average-Optimal-Shift-Or
APD	: Arabic Plagiarism Detection
ARM	: Several Association Rule Mining
ASMA	: Approximate String Matching Algorithm
AWN	: Arabic WordNet
BKDR	: The Brian Kernighan and Dennis Ritchie
CAR	: Class association rules
CMS	: Course Management System
DIFF	: German Distance Education Institute
DV	: Document Viewer
EBOM	: Extended-Backward-Oracle-Matching
EM Algorithm	: Expectation-Maximization Algorithm
ESMA	: Exact String Matching Algorithm
FJS	: Franek-Jennings-Smyth
FSBNDM	: Forward Simplified Backward Nondeterministic Dawg Match
GBPMA	: Greedy String Tiling Based Punctuation Matching Algorithm
GBTMA	: Greedy String Tiling Based Text Lines Matching Algorithm
GCC	: GNU Compiler Collection
GNU	: GNU's Not Unix!
GPL	: GNU General Public License
GSP	: Generalized Sequential Patterns
GST	: Greedy String Tiling
HASHq	: Hashing and g-grams
HTML	: Hyper Text Markup Language
HTTP	: Hyper Text Transfer Protocol
ITFS	: Instructional Television Fixed Service
KMP	: Knuth Morris Pratt
LBNDM	: Long Patterns Backward Nondeterministic Dawg Match
LCS	: Longest Common Subsequence

LMS	: Learning Management Systems
MOODLE	: Modular Object-Oriented Dynamic Learning Environment
MOSS	: Measure Of Software Similarity
NLP	: Natural Language Processing
PA	: Punctuation Arrays
PAN	: Plagiarism analysis, Authorship identification, Near-duplicate detection
PERL	: Practical Extraction and Report Language
PHP	: Personal Home Page
RARM	: Rare Association Rule Mining
RKR-GST	: Running-Karp-Rabin Greedy-String-Tiling
RKR-GST	: Running Karp-Rabin Greedy String Tiling
SA	: Stopword Arrays
SBNDMq	: Simplified Backward Nondeterministic Dawg Match q-grams
SIM	: The Software Similarity Tester
SMA	: Greedy String Tiling Based Stopword Matching Algorithm
SOV	: Subject-Object-Verb
SQL	: Structured Query Language
SSEF	: Streaming Single Instruction Multiple Data Extension Filtering
SSMA	: Semantic Sequence Matching Algorithm
TBD	: Informatics Association of Turkey's
TDK	: Turkish Language Association
TLA	: Text Line Arrays
TVSBS	: Thathoo-Virmani-Lakshmi-Balakrishnan-Sekar
VLE	: Virtual Learning Environments
VPL	: Virtual Programming Laboratory
WebCT	: The Blackboard Learning System
WMA	: Word Matching Algorithm
YAP	: Yet Another Plague
YTÜ	: Yildiz Technical University

1. INTRODUCTION

The rapid development and widespread use of online resources and information technology have contributed to the Internet becoming the world's largest source of information (Kaya and Özel, 2014a), an important tool in modern education used by millions of students every day. According to Maurer *et. al.* (Maurer *et. al*., 2006), "easy access to large databases and the Web, in general, have turned plagiarism into a serious issue for educational institutions". Incidents of student plagiarism have risen due to the Internet's widespread availability and use in academic settings (Rimer, 2003; Scanlon, 2003; Underwood and Szabo, 2003; Campbell, 2006; Comas and Sureda, 2008; Sunderland-Smith, 2008; Comas-Forgas and Sureda-Negre, 2010). The increased accessibility of the Internet and computers, and the ease with which electronic texts can be copied are among the factors that contribute to this problem.

Distance education, which is now also referred to as distance learning or e-learning, has existed for centuries. While according to North (1993), a few definitions even look to define it in terms of a single technology, according to Long-distance teaching (1990), others display distance education simply as a recent development of the class into a remote location (Spodick, 1995). However, such definitions are restrictive and fail to recognize the actual needs of distance education users. Ian Mugridge (1991) provides a better definition, describing distance education as "a form of education in which there is normally a separation between teacher and learner and thus one in which other means - the printed and written word, the telephone, computer conferencing or teleconferencing, for example - are used to bridge the physical gap."

One of the most useful distance education tools is Moodle[1], a free web-based learning management system. Instructors can use Moodle to create effective online learning communities and provide students with the information they need in order to be successful in the class. Lecturers can post class assignments, quizzes, notes, and

[1] http://moodle.org/

discussion forums to Moodle, giving the opportunity for easy online lesson delivery. Students, on the other hand, can use Moodle to participate in classes and submit their assignments to their lecturers from everywhere in the world (Kaya and Özel, 2014a).

The existence of Internet has advanced management systems, such as Moodle, international collaboration in education but, at the same time, has also raised the opportunity for student plagiarism (Shahabi, 2012). Access to information through Internet, search engines, and networks makes plagiarism an easy operation for students, especially for graduate students in universities, and might those taking grades without knowledge background (Menai, 2012). Some students use the Internet and distance education tools to plagiarize (Kaya and Özel, 2014a) when completing assignments and projects. Plagiarism detection and prevention have become difficult educational challenges to overcome.

There are many definitions of what constitutes plagiarism (Maurer *et. al.*, 2006). The word plagiarism is derived from the Greek word “plagion” and the Latin word “plagiarius,”. It means in English “kidnapper.” Plagiarism is defined as “the passing off of another person's work as if it were one's own, by claiming credit for something that was actually done by someone else” (Maurer *et. al.*, 2006). This term is described by the Georgetown University Honor Council (1999) as "the act of passing off the ideas or writings of another as one's own." According to the Council of Writing Program Administrators (2003), “Plagiarism occurs when a writer deliberately uses someone else’s language, ideas, or other original (not common-knowledge) material without acknowledging its source”. Hacker (2007) defines plagiarism as “Using another author’s intellectual property - language, visuals, or ideas - in your own writing without giving proper credit.”. When the aim is to create new knowledge (Garg and Singh, 2014), produce new ideas, or shed light on something from a different angle (Garg and Singh, 2014), plagiarism defeats the purpose of the academic environment. In this thesis, we think that if there is a similarity among text documents then, there is a probability that these documents are plagiarized. Plagiarism in textual documents is also frequent and detection of textual document plagiarism requires different tools than those found in source code plagiarism. Researchers have dedicated significant time to detecting academic textual

plagiarism. Many new tools have been developed over the past few years to help lecturers detect plagiarism in student assignments.

Detection of plagiarized documents becomes more difficult when there are a large number of assignments submitted by students. Automated plagiarism detection tools help to achieve this aim, but no tool is perfect. Tools and web sites[2] such as SIM, Sherlock, Text Matching Tool of Yıldız Teknik University, Document Similarity Tool of Yıldız Teknik University, Turnitin, iThenticate, WCopyfind, CopyCatchGold, Ferret, JPlag, Sherlock-Plagiarism Detection Software, Boss Submission System, CompareFilesX, DigiMarc Guardian, VeriGuide, Small SEO Tools, Plagiarism Detect, Plagiarisma.net, Glatt, Eve2, Viper, DocCop, Urkund, Plaggie , and Plag Tracker have been developed for textual plagiarism detection.

The above mentioned plagiarism detection tools use different algorithms to detect similarities between students' work and source codes or documents (Kaya and Özel, 2014a). Some algorithms measure the similarity between token sequences, while others compute the metric-based similarity between strings by constructing tiles, find similarity between Parse Trees, or calculate similarities between graphs. Some tools detect textual documents through string matching, bag of words and citation analysis (Garg, 2014). Other tools use a grammar-based method, semantics-based method, grammar-semantics hybrid method, external plagiarism detection method and clustering in plagiarism detection to detect textual plagiarism (Ali *et. al.*, 2011). One important algorithm that is employed for detecting similarities between source codes or documents is Greedy String Tiling Algorithm (Wise, 1993). It is based on tokenization of the source codes or documents. According to Porkodi and Lshivakumar (2012), the general definition of tokenization is "It is the process of breaking a stream of text up into words, identifiers, functions, phrases, symbols, or other meaningful elements called tokens". Mon (2012) states that "Tokenization is useful both in linguistics and in computer science, where it forms part of lexical analysis". Every language has different lexical structure and is independent of languages. Greedy String Tiling is proposed for comparing pairs of files to determine

[2] The web addresses of the tools are given in References section of this thesis.

the degree of their similarity. It is based on one-to-one matching, and is able to deal with transposition of substrings.

The aim of this thesis is to provide plagiarism detection for Turkish documents, namely to measure how similar two documents are to one other. While several programs have been developed for detecting and preventing plagiarism, many of these methods are proposed for source code plagiarism and the English language. Some online detection systems developed for plagiarism of textual documents written in English are also available for documents written in Turkish; however, these systems cannot fully detect the similarities when the adds of the words are changed or the synonym words are used. In this thesis, a system is planned to develop that it could perform the features which could not be fulfilled by these systems. Also the planned plagiarism detection tool should be included into the Moodle based distance education system for easy assessment of assignments.

In this thesis, we propose a system to detect plagiarism in textual documents in Turkish, specifically student assignments on Çukurova University's Department of Computer Engineering Moodle web site[3]. We aim to compare the similarities between textual assignments written in Turkish and loaded onto the Moodle distance education system and to detect plagiarism with a string matching algorithm. In the current Moodle software, there is not a free plagiarism detection module for Turkish assignments. We plan to develop a Greedy String Tiling based string matching algorithm by using a Turkish Stopword list, a stemmer, and a synonyms dictionary. This system will compare not only similarities of assignments in Moodle, but also similarities of all kinds of Turkish textual documents. We also aim to improve the results of the similarity detection by analyzing the web usage behavior of the students on Moodle.

The reason for the selection of this topic in this thesis study is the opinion that distance education will be used by many educational institutions in the future. As the importance of distance education is better understood every day, the need for automated plagiarism detection software is increasing. Manual evaluation is not feasible to check information systems that contain hundreds of assignment

[3] http://bmb.cu.edu.tr/moodle/bmb/moodle/

submissions. In many cases, human markers have differing opinions about particular submissions.

The proposed plagiarism detection system in this thesis is different in the fact that, before documents are compared, we use many dictionaries such as the Abbreviation Dictionary, the English-Turkish Glossary of Computer Terms, and the Turkish Thesaurus to improve similarity detection. In the proposed system, we use Greedy String Tiling and Cosine Similarity measure which are used by major successful existing plagiarism detection tools to compute similarity between textual documents that are uploaded to our Moodle system. There are only two commercial plagiarism detection tools which support Turkish documents, called Turnitin and WCopyfind. In this respect, this study fills the gap in the literature, and can result in academicians easily detecting student cheating. Also, it will draw attention to plagiarism in spite of the limited number of academicians who are working on this issue in Turkey. This study is necessary because there is not sufficient study for the Turkish. It has also examined that web usage mining and data mining could be used in plagiarism detection or not.

The remainder of the thesis is organized as follows: In Section 2, the previous work summarizes the literature about distance education, Moodle, plagiarism, automated plagiarism detection, and web mining. In Section 3, the material and methods used in our new plagiarism detection system are described. In Section 4, the results obtained from our new plagiarism detection system on test data are presented. In Section 5, conclusions and future research directions are given.

2. PREVIOUS WORK

In this section, the literature about distance education, Moodle, plagiarism, automated plagiarism detection, Turkish language, and web mining are summarized. This section also presents some of the significant previous work on plagiarism detection methods for Turkish language and usage of these methods on Moodle distance education system.

2.1. Distance Education

There is difficulty in finding a universal definition of distance education; "The ideas surrounding the educational endeavor are somewhat similar" (Keegan, 1996).

According to Rudolf Manfred Delling (1966), a German historian and bibliographer, "Distance education is a planned and systematic activity made up of the choice of teaching and supports of student learning. It acts as a bridge between student and teacher through at least one appropriate technical medium." (Delling, 1966).

To Dohmen (1967), former director of the German Distance Education Institute (DIFF) in Tubingen, "Distance education is a systematically organized form of self-study, made possible at a distance by means of media. In this case, the opposite of 'distance education' is 'direct education' or 'face-to-face education,' which takes place with direct contact between lecturers and students." (Dohmen, 1967).

According to Moore and Kearsley (1996), distance education is defined as "planned learning that normally occurs in a different place and requires a well-defined system of delivery that includes modified teaching techniques, alternative modes for communication, including, but not limited to technology, as well as alternative administrative and organizational components".

Lastly, Picciano's (2001) definition of distance education, as cited by Birnbaum, states that "distance education uses three current and popular forms of media; broadcast television, two-way videoconferencing, and asynchronous learning networks".

Any of the above mentioned definitions could potentially shape an emerging theory of distance education. It is also important to remember that as technology advances, it results in new ideas of distance education; however, the basic concept of distance education remains the same: the education of individuals in a non-traditional environment through various forms of media.

Many educational institutions have created solutions to their increasing educational needs through the development of distance education programs. Distance education allows educational paths to be determined by educators and students, who are separated with physical distance, using technology (audio, video, data and written text). It is a form of education in which students, teachers and teaching materials in different geographies are brought together through communication technology (Gündüz *et al.*, 2007). Using video, audio, active learning, simulations, and electronic advances appeals to a variety of students with multiple learning styles.

In general, distance education is collected under two main headings: ***i)*** synchronous and ***ii)*** asynchronous. Synchronous learning is a learning that requires an instructor and all students to be interconnected at the same time. The teacher's instruction is interactive and includes web conferencing, teleconferences, telecourses, and Internet chat sessions. The synchronous form of distance education is less flexible than asynchronous learning which does not require simultaneous participation of an instructor and all students in a given activity. In asynchronous learning teachers provide all the materials, lectures, tests or assignments to be accessed at a time convenient for students. The online courses, videotaped courses, and correspondence courses are some examples of asynchronous distance learning. Such a form of learning has many benefits for those who combine education with a professional work and family. A possible drawback here is that students must be self-disciplined and highly motivated to get the work done.

The Internet-based distance learning model can be defined as a transmission of educational content with the use of text, image, video, and audio files over the Internet online or offline. According to the Institute for Higher Education Policy (IHEP), the Internet-based distance education has gained a special status for three basic reasons. First of all, the Internet has become the predominant technology in distance education, due to its increasing telecommunications bandwidth capabilities. Secondly, the Internet-based distance education allows the teaching and learning process to happen "at any time and any place." Asynchronous interactive learning environments, especially, have become the signature characteristic of this field. Finally, the Internet-based distance education is, in many ways, fundamentally different than traditional classroom-based education hence attractive for learners (Merisotis *et al.*, 2000). The main difference is that the Internet-based distance education removes the physical barrier and time constraints for students and lecturers.

Internet-based distance education must have the following features (Al and Madran, 2004):

- The identification and management of users,
- Preparation of course content,
- Managing courses,
- Monitoring and analysis of student behaviors,
- Assessment of students' achievement status,
- The creation and management of interactive communication media.

2.1.1. Distance Education Tools

Computers and computer networks are rapidly becoming the preferred long-distance communication tool and are evolving as a major resource in distance education. The increasing communications capability of computers is being used in universities. There are many distance education tools that are presented in Table 2.1.

Table 2.1. The distance education tools and web addresses

TOOLS	**WEB ADDRESSES**
Adobe Connect	http://www.adobe.com/products/adobeconnect.html
AkademikLMS	http://www.akademik lms.org/
ANGEL	http://www.angellearning.com/community/higher_ed.html
ATutor	http://atutor.ca/
Avilar WebMentor	http://www.avilar.com/learning_management/lms.html
BlackBoard (WebCT)	http://www.blackboard.com/
Bodington	http://bodington.org/
Claroline	http://www.claroline.net/
CoursePark	http://www.coursepark.com/
Desire2Learn	http://www.d2l.com/
Dokeos	http://www.dokeos.com/
DotLRN	http://dotlrn.org/
Drupal	http://drupal.org/
eFront	http://www.efront.gr/
Enocta	http://www.enocta.com/ enocta/web/pdefault.aspx
eStudy	http://estudy.sourceforge.net/
ETUDES	https://myetudes.org/portal
Fle3	http://fle3.uiah.fi/
ILIAS	http://www.ilias.de/
IntraLearn SME	http://www.intralearn.com/
Janison Toolbox	http://www.janison.com.au/
Moodle	http://moodle.org/
OLAT	http://www.olat.org/
Perculus	http://www.perculus.com/
RCampus	http://www.rcampus.com/
Sakai	http://www.sakaiproject.org/
SimplyDigi	http://www.simplydigi.com/products.html

Among the others, Blackboard (WebCT) and Moodle are two most well-known web-based learning management systems used in universities and institutions of higher education (Kaya, 2012).

Early in 2006 owners of Blackboard Learning System and WebCT decided to join their forces and merge two companies under the existing name of one of them – Blackboard. The new entity continues to support both systems (Kaya, 2012). The Blackboard Learning System (WebCT) is a Web-based commercial distance

education system dedicated to education containing teaching resources and straightforward user hierarchy. It allows instructors to post course information and materials as well as readings and assignments. Thanks to the flexibility of the Blackboard Learning System (WebCT) it is easy to design a course curriculum or study schedules and continuations of education courses go on flawlessly. Not only can the teacher shift page layouts including font types or colors but also choose texts and icon links. It also facilitates interaction among users, who can have basic discussion, and offers other collaborative tools. The idea behind the Blackboard Learning System is to let teachers deliver course content, especially adjusted to large courses at lower levels. The role of an administrator of the system is limited to conducting basic operations such as initial course and teacher registration. The administrator is not overly dependent on it as many course management tasks, like student and course content registration or tests and statistics check, can be carried out by a teacher. The system consists of many communication and discussion features enabling active participation of students. Possibility of use of multimedia, an instructional option, is willingly utilized by teachers.

Moodle is an online course management system that is widely known and freely available. The word Moodle stands for "Modular Object Oriented Dynamic Learning Environment" and was created by Martin Dougiamas, a computer scientist and educator, who spent time supporting a Course Management System at a University in Perth, Australia (Cole and Foster, 2008).

Moodle is a software package that it is used to create Internet based courses and their Web sites. Moodle is used in 231 countries, providing support for 124 languages, and has 68,853,051 registered users according to Moodle Statistics[1] on the Moodle web site in November 2014[2]. There are 53,562 active sites that have been registered from those countries. Currently, 7,381,261 registered courses and 1,153,671 teachers are on Moodle. Moodle is utilized by both institutions and individuals. The list of the former is long, including universities, high and primary schools, governmental departments, military and healthcare organizations as well as

[1] http://moodle.net/stats/
[2] http://moodle.net/sites/

airlines or oil companies. Among individual users are homeschoolers, independent and special educators[3].

Moodle, a PHP-based open-source online learning system, has been used since 2002 as a distance education tool and has various versions supported by Windows, Linux, Unix, and Mac OS X operating systems. The latest version of Moodle is Moodle 2.8, released on 10 November 2014 (support for 2.8.x will end in May 2016).

Moodle is designed as a software package to help instructors create online courses, which are set up in modules. This type of software package is known as a Learning Management System (LMS) or Virtual Learning Environment (VLE). One of elements of the Moodle environment is a platform offering many valuable features: class schedulers, assignments, profiles of participants, chats, wikis. Among the features are also: an interactive glossary, e-mail, teacher-created lessons, quizzes, surveys and workshops. Chats, forums, wikis, and workshops make collaborative student work possible.

A number of programs, namely PHP[4], MySQL[5] and Apache[6], are required before the Moodle program is set up. PHP is a script language embedded into html codes which works in a server-side. An open-source code that is embedded into an html page is run on every visit and can be downloaded from the Internet. MySQL is a database management system that can run in the background and can respond to requests, such as a high performance Web server. The query language which is used to find the required data on MySQL Database is the SQL language. Apache is an open-source software Web server which is completely free and has a high performance.

Moodle is used as an open-source virtual learning environment because of its important features (Cole and Foster, 2008; Barnard *et al.*, 2007):

[3] http://docs.moodle.org/28/en/Usage
[4] www.php.net/
[5] www.mysql.com
[6] http://www.apache.org

- Ease of use for system admins, application developers, course designers, faculty, and students (Barnard *et al.*, 2007);
- Smooth, teacher-friendly design with increasingly useful capabilities;
- Easy-to-create electronic portfolios for each student;
- Rich sets of resource and activity creation tools (Barnard *et al.*, 2007);
- Permission of online/offline assignments, individual and group work, online assessment that can be graded or ungraded;
- Accessibility of free and unlimited downloads;
- Open-source and freeware;
- Large, active community of users who develop new features and enhancements;
- Availability of 3rd party modules to extend the student experience (Barnard *et al.*, 2007);
- Easy updates through its own version detection mechanism;
- Flexibility to add or remove functionality at many levels due to its modular structure;
- Database file repair and compatibility recognition;
- Secure system, ensuring data consistency and encrypted cookies;
- Suitable for a 100% online course or face-to-face education;
- Compatibility with simple web browser such as Internet Explorer, Google Chrome, or Firefox; and
- Categorical course search functions.

These three advantages –open source, social constructivism, and community mentioned features—make Moodle unique in the Course Management System space (Cole and Foster, 2008).

The other advantages of Moodle are[7]:

- Immediate and free access to important course information and material;

[7] http://docs.moodle.org/20/en/ Moodle_manuals

- Collaborative applications such as wikis, the journal, the database, and the glossary;
- Teachers from within a school can network resources;
- Additional learning tools, such as forums and quizzes (self tests);
- Easy marking; and
- To-the-minute tracking for document views, assignment uploads, and time spent on a forum.

2.1.2. Comparison of Blackboard & Moodle

Curriculum design is supported by all two systems by providing course templates thanks to which instructors can deliver course materials, define study schedules and plan class activities. The Blackboard Learning System (WebCT) ensures variety in course content and materials. In addition, The Blackboard Learning System (WebCT) assists students in their off-line efforts. Regarding communication and discussion, all two tools deliver discussion forums, chat rooms together with exchange of e-mails and files. The Blackboard Learning System (WebCT) also provides private folders and internal e-mail for students and give them an option of making their own notes. As for performance assessment, all two systems incorporate assessment and grading functions. Course administration is embraced again by all two tools by facilitating uploading of student data and course data in batches. The Blackboard Learning System (WebCT) is also equipped with direct data interfaces. It should be noted there are certain similarities between The Blackboard Learning System (WebCT) and Moodle like option of student enrollments in courses, access to discussion forums or taking quizzes and tests. Other common features are: file upload (e.g. Word, PowerPoint, audio), SCORM compliant, grading, course calendar and monitoring students' participation. Table 2.2 includes a comparison of BlackBoard and Moodle.

Table 2.2. Comparison of Distance Education Tools (Duzer and Munoz, 2005; Cole, 2005; Kozaris, 2010)

Feature	**Moodle**	**BlackBoard (WebCT)**
Bandwidth	Dial-up	Bandwidth hog
Cost	Free	Commercial
Discussion	Nested threads	Not nested threads
Tools	Blog, wiki, journal, glossary, workshop	Whiteboard
Customization	Open Source, so can change locally	Need to request change
User Stats	Chart Comparing students; number of visit per page	Time student spent on each page
Discussion Forum	Three types	One type
Upload and share documents	Yes	Yes
Online Chat	Yes	Yes
Lessons with paths	Yes	Yes
Online Discussions	Yes	Yes
Student peer review	Yes	No
Online Quizzes / Surveys	Yes	Yes
Online Gradebook	Yes	Yes
Student submission of documents	Yes	Yes
Student Journals	Yes	No
Embedded glossary	Yes	No

According to this comparison Moodle system was preferred in this thesis, because Moodle is free and open-sourced unlike The Blackboard Learning System (WebCT). The other reason is that Moodle has a very large community of people who develop new features and plugins. It has a student peer review unlike the others.

2.2. Plagiarism

The word plagiarism is of Greek origin ‘plagion’ and in Latin it is ‘plagiarius’, which means kidnapper (Garg, 2014). In early 17th century, the term plagiarism was adopted in English, which means “the practice of taking someone else’s work or ideas and passing them off as one’s own” (Oxford, 2013).

Plagiarism has many definitions, which is why there is not a single universal definition. According to the Council of Writing Program Administrators, plagiarism can be defined as “Plagiarism occurs when a writer deliberately uses someone else’s language, ideas, or other original (not common-knowledge) material without acknowledging its source” (Council of Writing Program Administrations, 2003).

There are different plagiarism methods such as copy/paste plagiarism, paraphrasing, translated plagiarism, artistic plagiarism, idea plagiarism, code plagiarism, word switch plagiarism, style plagiarism, metaphor plagiarism, file compression, authorship attribution, presenting the same content with other media images, video and text, improper use of quotation marks and misinformation of references (Lukashenko, 2007; Sandhya and Chitrakala, 2011). In source code plagiarism, someone tries to copy whole or the parts of source code written by someone else and pass it as their own (Ali *et al.*, 2011). In textual plagiarism on the other hand, someone produces a work piece by copying it in full or partially from an original source, namely a report, essay, scientific paper (Ali *et al.*, 2011).

In terms of software or source code plagiarism, Halstead (1977) metrics were used on Fortran programs in 1976. These metrics measured four values: number of unique operators, number of unique operands, total number of operators, total number of operands. Programs are considered to be plagiarized only when all four values coincide. After 1977, the systems of Donaldson *et. al.* (1981), of Grier (1981), of Berghel and Sallah (1984), or of Faidhi and Robinson (1987) introduced a much larger number of similarity metrics.

A modern concept of plagiarism has emerged in Europe in 18th century gradually becoming a serious issue especially in the last two decades with online plagiarism being easy to conduct on “copy and paste” basis. Legal systems try to guard an intellectual property as much as a physical one. However, the former is much more difficult to track. Ability to detect or prevent plagiarism is essential for academia and other sectors like music, media, publishing and finance.

Omnipotence of the Internet and the ease of its use make plagiarism a very tempting option chosen by many people who often do it without a moment of hesitation or a single thought dedicated to the fact it may be illegal. Fortunately, new software solutions have been developed ensuring an efficient method of fighting plagiarism based on the Internet resources search and matching found texts with the one subject to an investigation. Constant development of this technology gives hope that less and less people will be able to get away with an act of plagiarism.

Studies of student plagiarism have found clear differences between students' and teachers' thoughts on plagiarism. The most common factors of purposeful plagiarism are the stress to obtain good grades, inefficient institutional deterrents, and forgiving lecturers (Davis *et al.*, 1992). The awareness of plagiarism as a serious law breaking among students is at a low level (Franklyn-Strokes and Newstead, 1995). Students plagiarize to get a good grade (Ashworth *et al.*, 1997). As the reasons for such a state Zobel and Hamilton (2002) indicate mainly financial problems of students whilst Wilhoit (1994) claims that students entering a university don't understand the rules of academic writing. Howard (1995) argues that some students plagiarize simply because they have not been taught how to give proper credit to sources.

When students are in an environment where their peers are cheating and they are unlikely to be caught, the incidences of plagiarism increase (Ashworth *et al.*, 1997). But, when students feel as though they will be caught, academic dishonesty decreases (McCabe *et al.*, 2001). Even though it is clear that the responsibility for detection and fight with plagiarism lies on the shoulders of institutions. Zobel and Hamilton (2002) argue that majority of departments are not prepared sufficiently to deal with the problem as students in case of difficulties seek for their peers advice rather than consult tutors. A proactive community of teachers can lead to a decrease in academic dishonesty and students who avoid plagiarism.

2.2.1. Types of Plagiarism

Plagiarism in textual assignments is an unavoidable issue for most academic teaching programming. Articles, text books, scientific papers are the common sources, available in the Internet, used by students to obtain desired material. The ease of copying from such contents encourages students to plagiarize (Gomes, 2006). It is unveiled that the Internet is a tool for students to hire expert coders for textual assignment implementation (Gomes, 2006). Anti-plagiarism softwares, which are free-text plagiarism detection software and source-code plagiarism detection software, were evaluated by Bull *et al.* (2001) and Culwin *et al*. (2001) respectively.

Textual Plagiarism is when a piece of work is copied partially or entirely from one or more sources. Plagiarism is evident in a student's work when there is a sudden change of style from the student's norm, or the use of more advanced grammar and vocabulary. Materials that are plagiarized can contain unexplained acronyms or technical vocabulary described in an earlier section of the plagiarized document[8]. As the purpose of a plagiarized material was different, it is not to the point or un-encyclopedic in tone.

Textual plagiarism detection includes grammar-based method, semantics-based method, grammar semantics hybrid method and external plagiarism method. The grammar-based method facilitates detecting exact copies with no alterations. It examines grammatical structure of texts by string-based measurement of similarities between documents. In the semantics-based method similarities between documents are detected with the use of the vector space model. The grammar semantic hybrid method is for the natural languages. It is suitable for the copied text including text modified by rewriting. A reference corpus composed of documents from which passages might have been plagiarized enables the external plagiarism detection (Ali et. al., 2011).

Plagiarism detection is usually based on the comparison of two or more documents. There are different plagiarism methods including (Maurer, Kappe, & Zaka, 2006; Lukashenko, Graudina, & Grundspenkis, 2007; Oy, Kakkonen, & Cosma, 2010):

- ✓ Copy – Paste plagiarism (copying in exactly the same words);
- ✓ Paraphrasing (a restatement of a same text in different words);
- ✓ Translated plagiarism (content translation from one language to another);
- ✓ Artistic plagiarism (occurring in non-text media or presenting same work of someone else using different media: text, images, video etc.);
- ✓ Idea plagiarism (using similar opinions which are not common knowledge);
- ✓ Code plagiarism (using program codes and algorithms by copying from other's works without permission or reference);

[8] https://www.editage.com/files/Handbook-Ethical-Editing.pdf

- ✓ No proper use of quotation marks (failing to identify exact parts of borrowed content);
- ✓ Misinformation of references (adding made-up references or non-existing source).
- ✓ Forgotten or expired links to resources (addition of reference marks but failing to provide information or up-to-date links to sources)
- ✓ Autorship plagiarism (the direct case of putting your own name to someone else's work)

Paraphrasing plagiarism occurs when a writer summarizes an idea taken from another text without providing appropriate citation or a corresponding reference. Plagiarism of that type is not easy to prove. Paraphrasing can be considered as legitimate (Hexham, 1999):

1) When it doesn't dominate a work of a writer;
2) When it allows the writer a critical interaction with another person's views;
3) When the argument of an original writing is reflected with the use of different words.

Source Code Plagiarism is a form of academic dishonesty often committed by students in programming classes. In a large class, manual plagiarism detection poses serious difficulty. Millions of programming exercises are submitted every year by Computer Science students around the World. It is understood that students derive from each other exchanging their works via the Internet. Such operations may take place between students from different universities, countries or continents; therefore, even though instructors often suspect plagiarism they cannot prove it. In programming assignments, plagiarism does not only involve copying the source code, but also if someone includes comments, program input data and variable names (Iaqat and Aijaz, 2011). Generally, plagiarism in coding is difficult to detect because of the similar source codes used for the same programs. Students copy all or part of a

source code from different sources and submit the copy as their own work. This includes students who collaborate and submit similar assignments.

When a lecturer in a programming course gives same assignment to all students, then all students have to work on the same assignment. It is then possible that some students write source code on their own, while the other students take the code from them and change variable names and the order of statements and functions, and submit it. These types of modifications in the source code are very difficult to catch. There are two types of source code modification: ***i)*** Lexical change, and ***ii)*** Structural change. It is very easy to complete a lexical change. One can complete a lexical change without any knowledge of programming language by using a simple editor. Structural changes cannot be completed without any programming language knowledge (Goel *et. al.*, 2008; Hage *et. al.*, 2010).

2.2.2. Automated Plagiarism Detection

In the literature, there exist several studies which include plagiarism detection with/without distance education systems.

2.2.2.1. Plagiarism Detection in Natural Language Texts

In the study of Clough (2003), he focused on automatic textual plagiarism detection and its methods; aimed to present plagiarism detection as a problem to be solved and to measure text reuse in journalism. He explored the nature of the plagiarism problem, and in particular summarized the approaches used for its detection. He has described the types of plagiarism (copy - paste plagiarism, paraphrasing plagiarism, plagiarism of secondary sources, plagiarism of the form of a source, plagiarism of ideas and plagiarism of authorship). His opinion on the types of plagiarism was that the easiest type of plagiarism to detect was word-for-word plagiarism. He was discussed the approaches for detecting plagiarism in natural language texts and experimented with plagiarism detection methods to detect stylistic inconsistency across a single text in this study.

Maurer *et. al.* (2006) have presented a survey on textual plagiarism. They mentioned firstly about the general setting of plagiarism and the plagiarism detection tools, which are Turnitin, Mydropbox, Docol©c, Urkund, Copycatch, WCopyfind, Eve2, Glatt, Moss, and JPlag. Then they tested these tools and reported the results. According to the results, there is no evidence that any released plagiarism tool can detect paraphrased or translated plagiarism. The other finding is that the results of almost all tools and web services can not be used without human interpretation.

Butakov and Shcherbinin (2009) presented an approach to reduce the processing time and the number of search queries. They developed the default search parameters for a free plagiarism detection plug-in for Moodle using the modified Winnowing Algorithm during the similarity phase and the Microsoft Live Search service during the Internet search phase. The proposed method was language independent. The search query number was reduced and total time was decreased with this system, which performed with the random selection method. Five "copy-and-paste" plagiarized papers were tested. Each paper had about 550 words. The sliding window word number was seven. According to the results of the experiment, increasing number of search queries would not necessarily to increase the relevance of the results and 5% random selection was enough to find the source of plagiarized document.

In the other study of Shcherbinin and Butakov (2009), an approach for plagiarism detection was presented using Microsoft SQL Server platform. This approach used the Winnowing algorithm, a fingerprinting-based algorithm, and Levenstein distance has provided an advantage that it has a high precision, good performance and readiness. The proposed system could reduce the number of documents for comparison phase. The results were exported to XML files at the end of the comparison.

In the study by Telepovská and Gajdoš (2010), the problem of plagiarism was analyzed. A newly created system for plagiarism detection at Faculty of Electrical Engineering and Informatics, Technical University of Košice was described in this study. This system aimed to reduce the rate of plagiarism in the Faculty. Karp-Rabin algorithm and Greedy String Tilling algorithms were used to detect plagiarism. The

system can be extended to search for a comparison of identical and similar fragments in the source codes.

Chen *et. al.* (2010) implemented ROUGE (Lin, 2004), which was previously applied in the evaluation of text summarization, that uses WordNet[9], which is a large lexical database of English, to detect plagiarism. Rouge was capable of detecting verbatim copy. For the purpose of this study, the WordNet was used to detect synonyms and semantic-related words called unigrams. *N*-gram analysis up to 4-gram, LCS, skip-bigram and WordNet are the four methods of the system exploited to unveil plagiarism. Two manually generated data sets supported evaluation of the above mentioned methods. The first set was composed of 978 pairs of sentences. The second set, less impressive, contained 100 pairs of sentences. According to the experimental results, stemming was not effective on the plagiarism detection. Also, according to the results, ROUGE could detect the verbatim copies.

Jayapal, in his study (2012), reported the best performed approaches towards a candidate document retrieval task and a detailed comparison task for plagiarism detection prepared for PAN research community within PAN 2012 detection track. In this study, while the N-gram similarity overlap metric was used for candidate document retrieval task, for detailed comparison task the Greedy String Tiling algorithm was used. It was observed during the evaluation that the approach used for the candidate document retrieval task was highly competitive contrary to the approach used for detailed comparison task that turned out to require thorough improvement. The Greedy String Tiling (GST) Algorithm (Wise, 1995) was implemented for detection of plagiarized sections of the text in the detailed comparison task. The GST Algorithm having as a base Running Karp-Rabin Matching (Wise, 1993) was utilized in the study. To enable classification, the training data (Potthast *et al.*, 2012) provided for the detailed comparison task was used. Weka's (Witten *et al.*, 1999) Random Subspace Classifier based on the 10-fold cross validation results was also put to action to train and classify the document pair. Weka was used with the purpose of testing different classifiers. The F-Measure produced 0.791 values with the 10-fold cross validation across the training set with

[9] http://wordnet.princeton.edu/

the Random subspace classifier. The minimum tile length was varied and set for each class according to the plagiarism class identified by the classifier.

Jiffriya *et. al.* (2013) presented a fast plagiarism detection tool named AntiPlag using the tri-gram sequence matching technique for text based electronic assignments to reduce detection time. To reduce the time, a clustering approach was used on the detection and applied to the collected datasets to create appropriate clusters using K-Means algorithm using the WEKA tool. AntiPlag was tested and compared with Plagiarism Checker X tool. According to the results of the tests, the cluster-based AntiPlag showed better results and took less time than the Plagiarism Checker X. The results showed that AntiPlag was faster than Plagiarism Checker X for plagiarism detection on text-based electronic assignments.

Available tools and web services for textual plagiarism detection are presented in more detail in Section 2.2.2.4.

2.2.2.2. Plagiarism Detection in Turkish Texts

Işık and Çamurcu (2008) used the techniques such as Euclidean, Cosine, Pearson and Extended Jaccard to find document similarities in web pages. The aim was to cluster web documents. In the experiments completed for web documents clustering, they found that the Euclidean distance measure had high fault rates. The best performance in the similarity measures was provided by Cosine and Extended Jaccard measures. According to the results of the experiments, the Cosine similarity measure was found suitable to use in the web documents clustering.

In the study by Karadağ and Takçı (2010), a system which can classify similar news for a dynamic contents news site and scan the news in different categories and different sources were developed. This study's goal is to increase the functionality of searching and to show more beneficial information which are interrelated between each other for readers who search news in the web. Zemberek Turkish language processing library was used for the detection of word stems. To detect similarities, Cosine Similarity measure was used. According to the

experimental results, Cosine Similarity could detect similar texts, but it could find higher rate of similarity than actually existed.

In the study by Yücesoy (2007), a new semantic similarity calculation method was examined for Turkish documents. The proposed method was based on the Turkish lexical dictionary. This method was compared with existing lexical methods. Yücesoy (2007) focused on edge-based lexical methods, which is a lexical semantic similarity method. The edge based lexical method was consisted of the basic edge counting method and the Wu-Palmer method. It was stated that the Wu-Palmer method was more effective than the basic edge counting method, so they compared the Wu-Palmer method with the new lexical method. In their experiments, their new method outperformed the Wu-Palmer method. In this study, it was questioned whether or not to care for all semantic relations in taxonomy. Calculations with all relations made execution time higher. Additionally, utilization was not as big as execution time. Therefore, Yücesoy (2007) decided that considering all relations is unnecessary. Afterwards, it was just worked with IS-A relations in further experiments. Yücesoy (2007) decided to use cosine similarity for non-lexical methods. A graph-based clustering method was used since pairwise similarities were examined. The experimental results show that non-lexical methods performed better than the lexical ones, whether the dictionary was enough or not. To make this analysis easier, documents are clustered such that more similar ones were in the same group. According to the results, Cosine Similarity measure performed better than the Wu-Palmer and the proposed similarity measure.

Dursun (2008) improved a new similarity computing technique by modeling some cases often encountered in matching Turkish texts and computing similarities. Especially, an intelligent method has been developed that perceives slips in spelling, arranges them, and computes similarity. Here, “text” may be both a word composed of a few letters and a long text composed of hundreds of words. In order to measure the success of the method improved, different computer users who have different specialities, at different levels, were asked to type different texts. Then, these wrongly typed inputs were used. The percent similarity of the texts that users typed false and their correct forms were computed by three different methods: the

improved method, Edit Distance Similarity, and Jaro-Winkler Similarity. After that, their success was measured comparatively. According to test results, the improved method was more successful than Edit Distance Similarity and Jaro-Winkler Similarity. Also, a software that finds repeated or similar records on the tables in any Oracle database by using these three methods was developed.

2.2.2.3. Plagiarism Detection in Source Codes

In the study of Shcherbinin and Butakov (2008), an open-source anti-plagiarism module prototype based on the Winnowing Algorithm in Moodle course management system was developed. It compared student submissions (plain text, source code, essay, …etc.) from the Introduction to Computer Science course at American University of Nigeria in the 2006-2008 semesters with local document database as well as with results of global search returned by MSN Live™ search engine. Because of using some Microsoft technologies, the developed system could not used for Moodle plugin. This system had several servers for tokenization, so the type of submitted assignment was not important. The comparison of plagiarism found with manual grading and by a software was done. According to the result, the level of plagiarism found by software was higher than the manual grading. After the successful results, the desktop prototype of the plug-in was developed for processing of 1000+ submissions.

A standalone environment for managing assembly language programming assignments was also the subject of Rodriguez *et al.* (2007) efforts. Their work, however, hasn't resulted in adding new solutions to any existing distance education system. The tool developed by Rodriguez *et al.* executes a set of prior defined tests taken by students. Furthermore, students can debug their codes within the system. Their proposed system has a plagiarism detection assistant that completes the tool-chain functionality. Once the submission period has finished, the plagiarism detection assistant tool measures similarities between student submissions. Each submission was compared with four different similarity measures, the length of the larger substring, the length of the longest match, the percentage of the length of the

longest match, and the percentage of patterns common to both files. The need of Helsinki University of Technology to support large scale introductory programming courses was satisfied by an automatic assessment system authored by Malmi *et al.* (2002). At the heart of the system are TRAKLA[10] and Ceilidh (Foxley, 2004) systems for programming courses incorporating C, Fortran, Scheme and Java assignments. In a nutshell, the solution proposed by Rodriguez *et al.* (2007) relies on processing of source codes submitted by the students with the use of predefined test data sets. The system records the output that later is checked against a set of regular expressions defined by a teacher what aims at finding correct strings in the output. The described tool has no special modules dedicated to plagiarism detection. Plagiarism attempts are prevented by assignments' personalization.

Marques *et. al.* (2013) have presented an innovative academic approach regarding the use of PlanetLab, an open large-scale Internet platform used in engineering courses, on real distributed systems to run students' assignments. This platform provides automated feedback to students regarding the performance of their instances. PlanetLab has two main contributions, which are automatic assessment and the use of a real laboratory over the Internet. PlanetLab makes it feasible for students and educators to deploy, execute, and assess students' assignments. This environment has a Copy Detector and Automatic Assessment components. With Copy Detector component, it can detect possible plagiarism. The innovation here is to utilize JPlag to show similarities between plagiarized files.

CODESIGHT was a tool developed by Bejarano *et al.* (2013). It could recognize the programming source code similitude of student assignments particularly in those computer programming courses that were introductory. CODESIGHT was created basing on the Greedy String Tiling algorithm modification. Four tests, one theoretical and three real scenarios, were applied to determine the efficacy of this tool. According to these results, the percentage of similitude could be influenced by the language syntax and the size of the selected threshold.

[10] http://www.cs.hut.fi/~tred/WWW-TRAKLA/WWW-TRAKLA.html

2.2.2.4. Tools for Plagiarism Detection

Plagiarism detection programs are not new; in fact, they have existed for years. Some of them are used to detect text plagiarism; some of them are used to detect source code plagiarism. These tools are shown in Table 2.3 and 2.4.

Table 2.3. Free Plagiarism Detection Tools

Free Systems	**Type**	**Web Address**
WCopyfind	Textual	http://plagiarism.bloomfieldmedia.com/z-wordpress/software/wcopyfind/
CopyCatch Gold	Textual	http://www.cflsoftware.com/GoldFull.html
Ferret	Textual	http://peterlane.info/software/ferret.html
The Plagiarism Checker	Textual	http://www.plagiarismchecker.com/
Chimpsky	Textual	http://chimpsky.uwaterloo.ca/
CitePlag	Textual	http://citeplag.org/
CopyTracker	Textual	http://antiplag.sourceforge.net/
eTBLAST	Textual	http://etest.vbi.vt.edu/etblast3/
Plagium	Textual	http://www.plagium.com/
Anti-Plagiarism	Textual	http://ikc2.tup.km.ua/index_us.shtml
DupliChecker	Textual	http://www.duplichecker.com/
PaperRater	Textual	http://www.paperrater.com/
DocCop	Textual	https://www.doccop.com/
Plagiarisma.net	Textual	http://plagiarisma.net/
PlagTracker	Textual	http://www.plagtracker.com/
Viper	Textual	http://www.scanmyessay.com/
SeeSources	Textual	http://www.plagscan.com/seesources/
Plagiarism Detector	Textual	http://plagiarism-detect.com/
JPlag	Textual, Source Code	http://jplag.ipd.kit.edu/
SIM	Textual, Source Code	http://dickgrune.com/Programs/similarity_tester/
Sherlock	Textual, Source Code	http://sydney.edu.au/engineering/it/~scilect/sherlock
Moss	Source Code	http://theory.stanford.edu/~aiken/moss/
PMD Copy/Paste Detector (CPD)	Source Code	http://pmd.sourceforge.net/
YAP	Source Code	http://luggage.bcs.uwa.edu.au/~michaelw/YAP.html

Table 2.4. Commercial Plagiarism Detection Tools

Commercial Systems	**Type**	**Web Address**
Iparadigms: Ithenticate	Textual	http://www.ithenticate.com/
Turnitin	Textual	http://turnitin.com/
CheckForPlagiarism.net	Textual	http://plagiarisma.net/
PlagScan	Textual	http://www.plagscan.com/
Veriguide	Textual	http://veriguide1.cse.cuhk.edu.hk/portal/plagiarism_detection/index.jsp
SafeAssign	Textual	https://help.blackboard.com/en-us/Learn/9.1_SP_12_and_SP_13/Instructor/090_Assignments/030_SafeAssign
CopyCatch Gold (Commercial Edition)	Textual	http://www.cflsoftware.com/GoldFull.html
EVE2 Essay Verification Engine	Textual	http://www.canexus.com/
Glatt Plagiarism Screening Program and Service	Textual	http://www.plagiarism.com
Grammarly	Textual	http://www.grammarly.com/
Copyscape	Textual	http://www.copyscape.com/
Plagiarism Detect	Textual	http://plagiarism-detect.com/
University of Maryland Dustball Plagiarism Checker	Textual	http://www.dustball.com/cs/plagiarism.checker/
MyDropBox (formerly PlagiServe)	Textual	http://www.plagiserve.com/
Plagiarism Checker X	Textual	http://plagiarismcheckerx.com/
Docol©c	Textual	https://www.docoloc.de/
CodeMatch	Source Code	http://www.safe-corp.biz/products_codematch.htm
AC	Source Code	http://tangow.ii.uam.es/ac/
Plague	Source Code	http://www.freepatentsonline.com/y2010/0325614.html
Marble	Source Code	http://sydney.edu.au/engineering/it/~scilect/sherlock/
Plaggie	Source Code	http://www.cs.uu.nl/research/techreps/repo/CS-2010/2010-015.pdf
Siff	Source Code	http://www.cs.uu.nl/research/techreps/repo/CS-2010/2010-015.pdf
Bandit	Source Code	http://glimpse.arizona.edu/javadup.html

According to (Delvin, 2002; Lancaster and Culwin, 2000; Neill and Shanmuganthan, 2004; Lukashenko, 2007; The University Of Sydney Teaching And Learning Committee, 2003) the most known plagiarism detection tools are Turnitin, CopyCatchGold, Eve2 and Glatt. According to analytical information available on the Web the leader among the detection tools is Turnitin (IParadigms, 2006), due to its functionality. Application of each of tools is determined by a set of features.

Turnitin (Jokinen, 1996) uses Digital Fingerprint Creation, a document identifier made from a series of constantly changing, custom algorithms.

The Software Similarity Tester (SIM) (Grune and Huntjens, 1989), developed in 1989 by Grune in the Vrije University in Amsterdam, is a similarity detection program that can be used for C, Java, Pascal, Modula-2, Lisp, Miranda programming languages and text files. SIM runs locally. The system detects duplicated codes in large software projects as well as in the program text, shell scripts' documentation, and educational projects (Hage *et al.*, 2010). SIM can run on both Windows and UNIX/Linux by taking parameters from the console. SIM, a free and downloadable software can handle name changes and reordering of statements and functions and take into consideration the space characters (Arwin, 2006). It has the ability to import the comparison results to a text file (Özen, 2012). In combination with the option of SIM, it can set the threshold value above which similarities will be reported. Firstly, SIM tokenizes the source code. Then a forward reference table which can be used to detect the best matches between newly submitted files, and the text they need to be compared to is built. While SIM is no longer actively maintained and supported, its source code is publicly available. In the study by Hage *et al.* (2010), researchers compare five tools for detecting plagiarism in source code texts, including SIM. According to the study, the plagiarism detection tool seemed to be unique to VU University Amsterdam, and therefore, not very flexible (Hage *et al.*, 2010).

Sherlock is a program created by Pike which finds similarities between textual documents such as essays, computer source code files, and other assignments in digital form (Kaya and Özel, 2014b). It runs on both Windows and Unix/Linux. It will even work with Tar files, but not compressed archives such as Gzipped or Zipped files (in order to run on Zipped files it must unzip those files first). Sherlock is a command-line program which does not have a graphical user interface.

Sherlock is a system designed to use structure-based approach (Cosma, 2012). Digital signatures are applied to seek similar pieces of text. A digital signature can be described as a number formed by turning a sequence of input words into a series of bits that are later clustered back. Two programs: sig and comp are responsible for the operation. The sig program produces digital signatures storing

them in a file. The comp program takes over then comparing the signature files and reporting any similarities encountered. The two programs combined together created Sherlock. It let to save substantial space on a disc and increase management efficiency.

CopyCatch Gold has been used by individual university teachers for over ten years to monitor student work for plagiarism (CFL Software Limited, 2012). The paramount task of CopyCatch Gold is to track collusion; however, it has a broader application including checking if documents were produced independently or providing instant feedback to actions taken by course participants even if the class is large. The tool is suitable both for a classroom or one-to-one use. CopyCatch Gold is a fast, scalable and multi-platform program with multi-processor capability (CFL Software Limited, 2012). Initially written in only Java, it has been using Java, Hadoop, Jena (Rdf and Sparql) and Chapel since 2012. It works on files with rtf, doc, htm and txt extensions. It has a variety of settings at the interface available, including language and the similarity threshold. Results in the application interface are compared side by side and can be saved in html and rtf formats.

JPlag (Prechelt *et al.*, 2000; Prechelt *et al.*, 2002) is a system written in Java by Lutz Prechelt and Guido Malpohl of the University Karlsruhe and Michael Philippsen of the University of Erlangen-Nuremberg, to detect source code plagiarism (Zeidman, 2010). It supports not only C, C++ and Scheme languages, but also natural languages with .txt, .asc, and .tex extensions. Originally, JPlag found pairs of similar documents in the documents set. In 2005, JPlag was turned into a publically available web service (Hage, 2010) by Emeric Kwemou and Moritz Kroll, and now finds pairs of plagiarized programs among a given set of programs. It has been used successfully in practice to detect similarities among student programming assignment submissions. The program's feature highlights its accessibility and its unique interface for understanding results.

To use JPlag's free software, only an e-mail account is required. This web service can be accessed at any time with the username and password that is sent to the user. In order to use the service, the user supplies a directory of files on user's own computer. The files are uploaded to the remote server, and the server applies

JPlag to the files. It stores the resulting HTML pages to be viewed by a standard web browser. Only a particular JPlag user can view the web pages containing the results. Every source code or text part in the directory is later parsed and transformed into token strings. These token strings are next compared to each other with the use of Greedy String Tiling Algorithm (Wise, 1993), proposed by Micheal Wise, which takes a set of programs, compares these programs in groups of two and produces a set of side-by-side HTML pages in which similar segments of the code are marked with different font colors. The server holds web pages until their expiration within several days or overwriting with new submissions of the same user.

Ferret is a plagiarism detection tool developed by Lyon, Malcolm and Dickerson (2001) at the University of Hertfordshire which also detects duplicated text or code in pairs of documents or source files. It has both text and graphical user interfaces which are fast, convenient and easy to use. The graphical user interface has been rewritten using the wxWidgets library (Smart and Hock, 2005). It has been implemented in Java, C++, Visual Basic, Visual C++, and uses Microsoft Word macros, but has cross-platform compatibility. It runs with .rtf, .doc, .docx, .pdf, .txt, .abw, .as, .h, .c, .cpp, .java, .lisp, .php, .pl, .py, .rb, .vb, .xml, and .html documents. It has a graphical user interface, automated document download from Web, processing of programming language text, processing of Chinese language, and document conversion into plain text. Ferret counts the number of distinct triples of tokens common between documents. The standard template library is employed to write the algorithm in the C++ language (Musser, 2001). The system calculates a similarity measure which is based on trigrams detected within two documents compared. Numbers 0 – not copied and 1 – everything has been copied stand for the measure. It must be noted that the measure shouldn't be considered an absolute measure of copying amount. Ferret is useful for document analysis, tracking changes to or reuse of text in documents. It is both a standalone application and a web application. It includes document analysis, software developers, software evolution, teachers, and new material tracking. It highlights the similarities of all document comparisons, ranked by a similarity score, and saves the results to pdf or xml formats, for printing or further analysis.

In the study by Lane *et. al.* (2006), a demonstration of Ferret plagiarism detection tool is introduced. According to these authors, Ferret is a cleaner, simpler, and code-based tool. The graphical interface of Ferret shows how much of a document is copied.

Alzahrani *et. al.* (2009) presented a novel web-enabled tool, called Arabic Plagiarism Detection (APD), was presented specific for plagiarism detection in Arabic documents. Arabic Plagiarism Detection (APD) tool detects questionable contents in students' assignments and papers. It also marks and judges such work pieces. Another function of the tool is to educate students how to proceed correctly whilst using other peoples' ideas. APD was designed for integration with e-learning systems. As the name of the tool suggests it operates on Arabic scripts facilitating assignment submissions to a teacher within a classroom. It can also use the Internet and search Google to find similar resources. It can find the similar documents on the web by utilizing Google API. So this tool is intra corpus plagiarism detection tool. It fingerprinted each submitted document by taking the least-frequent 4-grams. Then APD tool compares the 4-grams to its intra-corpus document collection. At the last stage, a fuzzy-based information retrieval technique is used to detect plagiarism between documents.

In the study of Menai and Bagais (2011), a new content-based plagiarism detection tool, called APlag, was introduced for Arabic documents. APlag was consisted of preprocessing, (tokenization, stopword removal, stemming, and synonym replacement), fingerprinting and document representation phases. The Longest Common Substring algorithm was used for the similarity metric in APlag. Also, three heuristics based on document trees were used to detect similarities for each level in the tree and compared according to their common hashes in document representation stage. The defined levels of the tree were document level, paragraph level, and sentence level. In comparison with APD, the mean of recall and precision values are higher than APD. In comparison with Turnitin, APlag compared favourably in terms of quality of results (Menai, 2012).

Lukashenko *et. al.* (2007) discussed the plagiarism prevention methods and the plagiarism detection methods. They described the most widely used plagiarism

detection methods and analyzed the most known plagiarism detection tools (Turnitin, Eve2, CopyCatchGold, WordCheck, GlattMoss and JPlag). The analysis showed that simplicity is the major factor deciding about which of statistical metrics are implemented in tools. The other analysis of them showed that these plagiarism tools could not detect plagiarism as well as humans.

Gruner and Naven (2005) proposed a plagiarism detection tool. Their tool was language-sensitive and based on stylometry, a statistical method of analyzing a text to determine its author. For this tool, they were produced 62 measurements which could be applied to file pairs. They have used the Chi-square method for defining a notion of statistical significance. Two main issues related to plagiarism were taken into consideration in their tool case: different documents should be assigned to different authoring sources; different sub-documents of the same document should be authored by the same person.

Gitchell and Tran (1999) developed the design and implementation of a program called Sim to measure structural similarity between two C computer programs. Sim is useful for detecting plagiarism among a large set of source code files. This program is a direct application of string alignment techniques such as those used in DNA string matching. Sim deals with source programs converting them into token strings and later comparing the strings. The technique utilized here is the dynamic programming string alignment technique.

In the study by Shahabi (2012), three plagiarism detection tools—Ferret, Sherlock and Turnitin— were compared with each other in an extrinsic plagiarism detection experiment. This experiment included documents classified into three types of plagiarism (near copy, light revision and heavy revision) and one type of non-plagiarism. The statistical methods of n-gram words match between texts detect plagiarism (Shahabi, 2012). According to the results, Ferret and Sherlock produced the same results in plagiarism detection performance in most cases, while Turnitin produced a higher percentage of similarity between the documents and the source.

In the study by Hage (2010), five source code plagiarism detection tools, namely JPlag, Marble, Moss, Plaggie, and SIM were compared with respect to their ten tool features and performance. They compared the performance using sensitivity

analysis and the comparison of the top-n output. According to the results, many tools were sensitive to numerous small changes. The top-10 results for JPlag, Marble and Moss were fairly similar, but Plaggie and SIM differed from the other three.

Riad *et. al.* (2013) presented a study of different plagiarism detection methods and types. They described common feature of these systems, which are Turnitin, APlag, Eve, APD, YAP, JPlag, SID and Moss. A web based system called as ZPlag was presented in this study to detect plagiarism in different fields such as documents, source code, images and e-learning systems. While in source code detection, the tokenization technique was used, the similarity algorithm was employed for detecting plagiarism in documents and the image vector features representation was applied for detecting plagiarism in images.

In the study of El Tahir Ali *et. al.* (2011), an overview for effective textual plagiarism detection tools was made. These selected tools which are PlagAware, PlagScan, CheckForPlagiarism.net, iThenticate and PlagiarismDetection.org were used for natural language text plagiarism detection, external plagiarism detection, and clustering-based plagiarism detection and compared with each other. According to the comparison, no software can detect plagiarism 100% because of their limitations. They ranked these tools starting from the best as follows, PlagAware, iThenticate, PlagScan, CheckForPlagiarism.net and PlagiarismDetection.org.

Mozgovoy (2006) considered the solutions of modern plagiarism detection programs. He mentioned the definition of plagiarism, the online and offline plagiarism detection systems and autorship identification in his study. He categorized the offline plagiarism detection tools into three categories such as fingerprint-based, string-matching-based, and tree-matching based systems.

There are many textual plagiarism detection tools and web sites for many languages but for Turkish there are only limited number of tools and web sites such as Turnitin, iThenticate, and WCopyFind.

WCopyfind[11] is a software developed by the University of Virginia in 2002. WCopyfind, which shares a common core code for loading, hashing, and comparing documents, provides a windows-style graphical user interface. It makes comparisons

[11] http://plagiarism.bloomfieldmedia.com/z-wordpress/software/wcopyfind/

using the documents which are available on the computer. This program can help in this regard if it was quoted from online works or if the writer forgot to reference the work. WCopyfind, licensed under the Gnu Public License, is free and available to anyone and supports the Turkish language. It reads .doc, .docx, .txt, .pdf, .htm and .html files; however, it cannot search the web or Internet to find matching documents. Some settings are presented to the user in the program interface, such as removing the upper/lowercase letters, punctuation marks, numbers and other characters; shortest phrase to match, the minimum percentage of matching words, and language. It performs an analysis in two steps. In the first step, the program loads hash-codes all of the documents. Words less than four characters are not examined. In the second step, each word is converted into a 32-bit hash code, over which the comparison process is performed. Matching phrases are underlined in a resulting html report. This program allows to users to see the two files on the same page side-by-side in the generated report. At the same time, the common words among texts are reported to the user as a txt file.

Turnitin, created by iParadigms, LLC in 1997, is a tool teachers can use to check student work for originality. Rather than simply a plagiarism detection tool, David and Carroll (2009) emphasize that it can be used to help students avoid plagiarism. Originality Reports can help teachers to locate potential sources of plagiarism. Turnitin also includes marking and grading features, which are GradeMark digital assessment, PeerMark, and GradeBook that can be used to improve academic writing skill. If used as a formative assessment tool, teachers can use Turnitin to point out where students can improve their citations, paraphrasing, and writing in general. After an assignment is submitted, in a search of text matches, it is compared against the gigantic data gathered in the Internet, but also against the information stored in local repository, previously submitted papers, journals, and publications. Turnitin cannot identify incidents of plagiarism or prove that a student has not plagiarized. It is still up to the teacher to determine if the work has been plagiarized. The result of this comparison, known as Originality Report, gives details about the matches and shows the degree of similarity between a submitted assignment and sources of content within the database. If a match between a

submitted piece of work and an existing work is found, Turnitin will highlight the matching text and provide a summary on the Originality Report. The icon of Originality Report displays a percentage and a corresponding indicates an index to which this percentage falls in terms of matching content. The above mentioned percentage is the Similarity Index. The higher the Similarity Index is, the greater the amount of text in the submission matches information gathered in Turnitin's database.

Turnitin uses a matching algorithm to find strings of words within assignments that are identical to those within its repository. This means that human judgment, based on various factors and considerations, is required to determine if an instance of plagiarism has occurred. Its specifically designed algorithms are used to create a digital fingerprint of any text. While Turnitin is good at searching the web and archives of old web pages, and serving as a warning to students, it is not good at invisible Web, print sources, one match, and interpreting proper citation.

Turnitin currently accepts .doc, .docx, .wpd, .html, .txt, .ps, .rtf, and .pdf file types for upload into an assignment. Text only files may not exceed 2 MB, but the other files may not exceed 20 MB. Large-sized files can be reduced by erasing non-text content. Turnitin does not allow uploading files that are password protected, encrypted or hidden. Likewise, system or read-only files are not accepted. The limit of zip file upload is set at 1000 files or 200MB of zipped data. It is advised to check the zipped files formats before uploading. Turnitin integrates with more than 50 Course Management Systems, including Blackboard, Desire2Learn, Instructure Canvas, Moodle, Pearson Learning Studio, and Sakai. Turnitin supports 19 languages: Arabic, Chinese (Traditional and Simplified), Dutch, English, Finnish, French, German, Italian, Japanese, Korean, Polish, Portuguese, Romanian, Russian, Spanish, Swedish, Turkish and Vietnamese. Turnitin will only match text between text of the same language.

iThenticate, launched in 2004 by iParadigms, LLC, is a corporate market plagiarism detection service, which also runs Turnitin and Plagiarism.org. It is designed to be used by institutions rather than individuals, but lastly they provided a limit service for single plagiarism detection user like master and doctoral students.

This allows the program to check a single document of up to 25,000 words, which can be used to check citations and originality in thesis drafts and other assignments. iThenticate is a plagiarism detection software that has two paramount benefits. Firstly, authors can make sure they have cited their sources correctly and their written work is of the highest quality. Secondly, this preliminary editorial review facilitates viewing and moving of submitted documents through the peer-review or referee process to publication, confident that content is original. iThenticate currently accepts the following file types for upload into an assignment: MS Word, XML, WordPerfect (WPD), PostScript, PDF, HTML, RTF, Hangul HWP, OpenOffice (ODT), and plain text. Files cannot exceed 400 pages, 40MB, or 2MB of raw text. Zip files may contain up to 200MB or 1,000 files. The iThenticate interface currently supports English, Korean, and Japanese. iThenticate searches for content matches in the following 30 languages: Arabic, Chinese (simplified and traditional), Japanese, Thai, Korean, Catalan, Croatian, Czech, Danish, Dutch, Farsi, Finnish, French, German, Greek, Hebrew, Hungarian, Italian, Norwegian (Bokmal, Nynorsk), Polish, Portuguese, Romanian, Serbian, Slovak, Slovenian, Spanish, Swedish, Russian, and Turkish. iThenticate will only match text between texts of the same language. iThenticate has a large database, and the possessed information is matched with the document in question. Highlighted text will include text that has been properly quoted and cited so cannot be qualified as plagiarized. This function makes possible to identify and attribute any material containing unintentional plagiarism. But multiple document comparison feature is not active that it cannot be used in iThenticate. Synonym & Sentence Structure Checking are not supported by iThenticate.

The main features of iThenticate are[12]:

- Database Checking
- Internet Checking
- Publications Checking
- Multiple Document Comparison (Currently not active)

[12] http://plagiarism-checker-review.toptenreviews.com/index.html

iThenticate offers reports, called the Document Viewer (DV) Report and the Similarity Report. iThenticate Similarity Reports provide an overall similarity index for each submission. The index computes the similarity level in percents between submitted text and information existing in the iThenticate databases selected as search targets. Delivery of a document's original format, including images, tables and graphs, within the Similarity Report, is ensured by The Document Viewer Report. It can be viewed for a Similarity Report in four different modes: Match Overview, All Sources, Match Breakdown, and Side By Side Comparison.

In the study of Kaya and Özel (2014b), the performance of plagiarism detection tools including CopyCatchGold, Sherlock, SIM, WCopyFind, JPlag, Text Matching System and Document Similarity developed by YTÜ Kemik Group, their implemented Cosine, Dice, and Jaccard measures were compared on Turkish sample datasets. The effects of using Turkish character set and Turkish stemmer on plagiarism detection were investigated. According to the experimental results, the most successful tools are Sim, WCopyfind, CopyCatch Gold, and their implemented Cosine Similarity. Also, these results showed that using Turkish characters decreased similarity detection, and using stemmed words increased the performance of plagiarism detection tools.

The comparison between the textual plagiarism detection tools and web systems are summarized in Table 2.5 and 2.6 in Appendices.

2.2.2.5. Plugins and Tools for Moodle

Gutiérrez *et al.* (2009) improved Moodle so it could check and verify automatically VHDL-based designs of the students. This extension was named CTPracticals. Among the functions of the extension there are automatic computer organization assignments grading and feedback message sending to students. CTPracticals ensures plagiarism prevention thanks to use of team identifier of the student groups.

Ramos *et al.* (2011) saw a potential in CTPracticals and decided to use it in a programming course organized by their Chemical Engineering Department. It

resulted in use of the extension for automated Matlab programming assignment evaluation. In addition, their Moodle extension can evaluate the students' Matlab codes and automatically generate feedback messages. From a questionnaire prepared by Ramos *et al.* (2011) and filled in by students it could be learned that the system was evaluated as useful and the success rate at the course increased. The plagiarism prevention was, in this case, ensured with file names as a function of user's identifier.

Jarpeb system is a proposal from Spinellis *et al.* (2007). It automatically assesses Java programming language assignments. The system operates on local computers of students but is not combined into a distance education system such as Moodle. Jarpeb aims at generating randomized programming assignments for each student to avoid plagiarism and to provide automated feedback and grading of the programming assignment submissions of the students.

Tresnawati *et. al.* (2012) have worked out a design basing on a plagiarism detection system model for programming assignments in a virtual classroom. The detection system implements Moodle and its delivery system alerts for students. The software detects plagiarism in assignments informing about a percentage of similarity between given files. It generates plagiarism detection alerts and delivers them to students with the use of SMS Gateway – an application facilitating receiving and sending messages.

Kaya and Özel (2014) included an online compiler and a source code plagiarism detection tool into the Moodle based distance education system of our Computer Engineering department. The aim was to extend Moodle system with the GCC compiler to allow lecturers to compile and run the students' source codes within the Moodle system, and the Moss source code plagiarism detection tool to detect plagiarism for programming assignments. According to the experimental results, using the online compiler and the plagiarism detection tool reduced time and effort needed for the assessment of the programming assignments; prevented the students from plagiarism; and increased the success of students in their programming course.

The Moodle Community has also made several efforts to include plagiarism detection facility to the Moodle system. Plagiarism detection plugins have been developed for Moodle to detect textual plagiarism including Urkund[13], Turnitin[14], Crot[15], Crot Pro[16] and Compilatio[17]. These tools except Crot are commercial tools for Moodle. Crot is a free plagiarism detection tool, but it cannot work with all versions of Moodle. This standalone tool is language independent. It is based on document fingerprinting. The textual documents can be analysed with Urkund against the content of three source areas including the Internet, Published Material and Student Material. After completion of the analysis, the generated reports are created for lecturers. Crot Pro is only being performed on English Wikipedia. It can compare documents side by side. Compilatio can compare documents from the same class or the class of the previous year. It can compare .txt, .pdf, .rtf, .doc, .docx, .odt, .xls, .xlsx, .ppt, .pptx, .html, .php, and .asp file types.

For source code plagiarism detection, Source-Code Plagiarism Plugin[18] which includes Moss and JPlag plagiarism detection tools have been proposed. The Source-Code Plagiarism Plugin only works with Moodle 2.0 and later versions.

There is another effort called Moss Plugin[19] to include Moss into the Moodle system. However, this plugin is currently unsuitable to use as the downloadable version of this plugin is not included on the web site. After the plagiarism settings were made, there was no way to display or calculate the results, even after we wrote the web address of the page which shows the results.

Virtual Programming Lab (VPL)[20] is another module that allows for the management of programming assignments with functions to upload, download and view of assignments, backup and restore data, grade, access control based on roles, and access logs (Chan *et al.*, 2004). With VPL, which is installed as a separate activity, users can examine the assignments. By clicking the 'Submission List'

[13] https://moodle.org/plugins/view.php?plugin=plagiarism_urkund
[14] https://moodle.org/plugins/view.php?plugin=plagiarism_turnitin
[15] https://moodle.org/plugins/view.php?plugin=plagiarism_crot
[16] https://moodle.org/plugins/view.php?plugin=plagiarism_crotpro
[17] https://moodle.org/plugins/view.php?plugin=plagiarism_compilatio
[18] https://moodle.org/plugins/view.php?plugin=plagiarism_programming
[19] https://moodle.org/plugins/view.php?plugin=plagiarism_moss
[20] https://moodle.org/mod/data/view.php?d=13&rid=4042&filter=1

button, the assignments and the names of students who send assignments appear in a list. Clicking on the Submissions button displays the size in bytes and the number of rows. The virtual machine, Jail Server, is required for executing the source codes of assignments. VPL allows users to edit and execute programs and enables automatic and continuous assessment, but it cannot allow using input files. VPL allows for the searching of similar files so it can control the authorship of the file. The system may add water marks in the downloaded code. It allows limited editing and prevents external code to be pasted. VPL edits, runs, debugs, and evaluates source codes exploiting a code editor. It supports jail servers ensuring a safe environment for compilation and running source codes. Jail servers serve to conduct interactive and non-interactive executions of source codes. Plagiarism detection algorithm contained by VPL works basing on comparison of the signatures in source files. The latest version of VPL module is dated September 2011 with several updates launched ever since. On the side of drawbacks, VPL is not friendly for users as it is complex and not supported with an operating manual.

2.2.2.6. Algorithms for Plagiarism Detection

A novel method for detecting monolingual plagiarized passages in document collection was presented by Stamatatos (2011). This method is an algorithm which detects exact passage boundaries. A modest list of stopwords is the base for this method. Thanks to the list Stamatos (2011) was able to isolate structural information, unlike the existing approaches, which are based on content terms to represent documents. He used the stopword *n*-gram method as it is utilized to determine similarity in the document level. This method is required minimal resources and text pre-processing cost.

In their study, Su *et. al.* (2008), focused on the Levenshtein distance use to change the likely scarcity and Smith-Waterman algorithm application for local similarities. In a consequence they described a hybrid plagiarism detection method. Su *et al.* (2008) utilized a diagonal line derived from the Levenshtein distance to decrease the scarcity of the dynamic programming array in the plagiarism detection.

Their method avoided globally involved string comparisons. They first used the Levenshtein Distance to divide the table, and then they applied Smith-Waterman Algorithm to the rest of the table. They implemented three different programs that are general, Levenshtein distance, and Smith-Waterman algorithm. According to the results, the proposed method is powerful in detecting plagiarism in practice.

Wise (1995) is known for developing a system called Neweyes. The system incorporates a novel string matching algorithm - RKR-GST, an acronym for Running Karp-Rabin Greedy String Tiling, to align nucleotide or amino acid bio-sequences. It enabled different operations to happen simultaneously: tiling one string with matching substrings of a second string. Transposed substrings or substrings of one bio-sequence that appeared rearranged in a second sequence could be recognized basing on the algorithm performance. Detection could also embrace repeated substrings.

In the study by Fredriksson and Navarro (2004), a new algorithm for multiple approximate string matching was presented. This algorithm is based on a sufficient backward reading of ℓ-grams of text windows, in order to show that no occurrence can contain the part of the window read, and then shifting the window. Four contributions were made to this study. The first contribution was to fill a significant gap in the area, because there is no average-optimal algorithm suitable for multiple approximate string matching. The second output was to give a practical algorithm for this problem, much better than already known alternatives in many cases of interest. The third contribution was to enable to increase the difference ratio so it could be higher than the algorithm of Chang and Marr (1994). The fourth output was to be the fastest at low rates differentials and moderate size of the alphabet and answer affirmatively to the question if practical average-optimal algorithm for approximate string matching existed.

Cantone *et. al*. (2012) interested in compressed string matching problem on Huffman encoded texts resulted in a new approach based on the Boyer-Moore strategy. A candidate's valid shift location triggers verification phase that tracks if the shift is codeword aligned. All of this happens thanks to use of the skeleton tree data structure. The approach is strongly linked to the algorithms exhibiting a

sublinear behavior on the average. The results of the experiments indicate that the algorithms exhibiting a sublinear behavior on the average skip more than 50% of the total number of bits in the encoded text.

In the study by Carus *et. al.* (2007), string matching algorithms were analyzed in terms of time depending on the length of the string, and the alphabet used. In this study, 13 algorithms were selected with different features. According to this study, performances of the algorithms vary on the length of the alphabet and string.

Ersin (2008) examined pattern matching algorithms, determined their efficiencies over different alphabets, and queried the effects of structural differences of natural languages over pattern matching algorithms. In addition, a fast pattern matching algorithm, which depends on the group frequency of alphabet characters is presented. According to the tests, the most efficient algorithms are Shift Or for DNA alphabets, Berry-Ravindran for numeral alphabets, and Tuned Boyer Moore for natural language alphabets. The worst algorithms are Reverse Factor for DNA alphabets and Forward Dawg Matching for both numeral and natural language alphabets. According to the test results of eight different natural languages, a pattern matching algorithm performance is affected not only with the total number of alphabet characters but also with structures of natural languages. The new pattern matching algorithm that is presented in this study, which can make searching by grouping the alphabet characters and perform string matching on natural languages, had shown performance on natural languages better than other effective pattern matching algorithms.

In the study by Faro and Lecroq (2013), forty seven string matching algorithms which were developed between year 2000 and 2010 were presented. After year 2010 several experiments were conducted within the field extending the existing knowledge. All algorithms had been implemented in the C programming language and compared with 12 texts of different types. The patterns were divided into four classes according to their length and the alphabets were divided into 4 classes according to their size. According to their experimental results, 11 algorithms were found as the best performers. Among these 11 algorithms, 6 of them are bit-parallel algorithms namely Simplified Backward Nondeterministic Dawg Match-Boyer-

Moore Horspool/Boyer-Moore Horspool- Simplified Backward Nondeterministic Dawg Match (SBNDM-BMH/BMH-SBNDM), Long Patterns Backward Nondeterministic Dawg Match (LBNDM), with Simplified Backward Nondeterministic Dawg Match q-grams (SBNDMq), Forward Simplified Backward Nondeterministic Dawg Match (FSBNDM), Average-Optimal-Shift-Or (AOSO) and Shift-And (SA), 4 of them are comparison based algorithms that are Thathoo-Virmani-Lakshmi-Balakrishnan-Sekar (TVSBS), Hashing and q-grams (HASHq), Franek-Jennings-Smyth (FJS) and Streaming Single Instruction Multiple Data Extension Filtering (SSEF) and one of them is automata-based algorithms Extended-Backward-Oracle-Matching (EBOM).

2.3. String Matching Algorithms

String Matching is one of essential subjects in the field of text processing. Practical software operating under majority of systems utilizes string matching algorithms. String Matching algorithms deliver challenging problems and through this plays an important role in theoretical computer science (Kosla and Blockeel, 2000). String matching algorithms can be divided into two groups: approximate string matching and exact string matching.

2.3.1. Exact String Matching Algorithms

One of the frequently studied topics linked to the computer science is the Exact String Matching problem. The importance of the problem is related to its direct application to such diverse areas as speech analysis and recognition, text, image and signal processing, data compression, information retrieval, computational biology and chemistry (Faro and Lecroq, 2013). The majority of large number of exact string matching algorithms is concerned with how to slide the pattern. Some of the exact strings matching algorithms are Boyer Moore, Tuned Boyer Moore, Karp-Rabin, Morris Pratt, Knuth-Morris-Pratt, Quick Search, Brute Force, String Matching on Ordered Alphabets, Deterministic Finite Automaton, Shift Or, Maximal Shift,

Simon, Colussi, Not So Naïve, Galil-Giancarlo, Apostolico-Crochemore, Apostolico-Giancarlo, Turbo-BM, Reverse Colussi, Horspool, Zhu-Takaoka, Berry-Ravindran, Reverse Factor, Turbo Reverse Factor, Smith, Raita, Forward Dawg Matching, Backward Nondeterministic Dawg Matching, Backward Oracle Matching, Galil-Seiferas, Two Way, Optimal Mismatch, Skip Search, KMP Skip Search, and Alpha Skip Search.

2.3.1.1. Karp-Rabin Algorithm

The Karp-Rabin Algorithm, created by Michael O. Rabin and Richard M. Karp in 1987, is an effective string searching algorithm which uses a hash function. The hash technique of Karp-Rabin algorithm is used to detect any set of patterns in a string (Dunham, 2003). Computing hash(x) is the base for the preprocessing phase of the Karp-Rabin Algorithm. It can be done in constant space and $O(m)$ time (Charras and Lecroq, 1997).

2.3.1.2. Morris Pratt Algorithm

Tight analysis of the brute force algorithm stands for a base for the Morris-Pratt Algorithm (Getoor, 2003). The algorithm performs comparisons from the left and moving to the right. The important factor here is the examination of the way the information gathered during a scan of text is wasted. The searching phase can be completed in $O(m+n)$ time. In the searching phase conducted by the Morris-Pratt algorithm it compares text characters. The maximum capacity of the performed comparisons is expressed with the $2n$-1 figure. The delay understood as the maximum number of comparisons for a single text character is bounded by m (Charras and Lecroq, 1997). m is the length of the first string, and n is the length of the second string.

2.3.1.3. Boyer-Moore Algorithm

The Boyer-Moore Algorithm (Charras and Lecroq, 2004) is a string searching algorithm which performs comparisons from right to left. The preprocessing phase of Boyer-Moore Algorithm is in $O(mn)$ time and space complexity and the searching phase in $O(mn)$ time complexity (Charras and Lecroq, 1997). Maximum $3n$ text character comparisons are conducted when searching for a non-periodic pattern. The Boyer-Moore algorithm incorporates two pre-computed functions: the good-suffix shift (the matching shift) and the bad-character shift (the occurrence shift). The functions are triggered in case of a mismatch or a complete match of the whole pattern and move the window to the right. The best performance of the comparisons are $O(n/m)$ (Charras and Lecroq, 1997). m is the length of the first string, and n is the length of the second string.

2.3.1.4. Quick Search Algorithm

The Quick Search Algorithm (Charras and Lecroq, 2004) is a simplification of the Boyer-Moore algorithm. The Quick Search Algorithm uses only the bad-character shift table. The pre-processing phase of this algorithm is in $O(m+ \sigma)$ time and $O(\sigma)$ space complexity (Charras and Lecroq, 1997). The searching phase of this algorithm is in $O(mn)$ time complexity (Charras and Lecroq, 1997). m is the length of the first string, and n is the length of the second string. The compared elements are patterns and text characters. The searching phase in which the comparison takes place is characterized by a quadratic worst case time complexity and good practical behavior. Comparing attempts can be conducted in any order. This algorithm gained recognition for being very fast in practice when applied to short patterns or large alphabets (Charras and Lecroq, 2004).

2.3.1.5. Zhu-Takaoka Algorithm

The Zhu-Takaoka Algorithm (Charras and Lecroq, 2004) is a variant of the Boyer-Moore Algorithm. This algorithm shifts the window after consideration of the bad-character shift for two consecutive text characters (Charras and Lecroq, 1997). The time and space complexity of preprocessing phase is $O(m+\sigma^2)$. The searching phase has a quadratic worst case which is *O(mn)* time complexity (Charras and Lecroq, 2004).

2.3.1.6. Berry-Ravindran Algorithm

The Berry-Ravindran Algorithm (Charras and Lecroq, 2004) is a hybrid algorithm of the Quick Search, Zhu, and Takaoka algorithms. The algorithm considers a bad-character shift for two back-to-back text characters immediately to the right of the window (Charras and Lecroq, 2004). The pre-processing phase of the algorithm is in $O(m+ \sigma^2)$ space and time complexity. The searching phase of the Berry-Ravindran algorithm has a $O(mn)$ time complexity (Charras and Lecroq, 2004).

2.3.1.7. Optimal Mismatch Algorithm

The Optimal Mismatch Algorithm (Charras and Lecroq, 2004) is a variant of the Quick Search Algorithm. In this algorithm, the pattern characters are scanned beginning with the least frequent one and finishing with the most frequent one. It requires the frequencies of the character and can quickly scan an entire text. Optimal Mismatch is a string matching algorithm which in the first step compares the rarest character in a text in decreasing order. Then it consist in building the Quick Search bad-character shift function and a good-suffix shift function adapted to the scanning order of the pattern characters (Charras and Lecroq, 2004), so it means that when the character doesn't match, the next character in the text, beyond the search string, indicates where the next possible match begins (Prasad and Panicker, 2010). It can be

done in $O(m^2+\sigma)$ time and $O(m+\sigma)$ space complexity. The searching phase of the Optimal Mismatch algorithm has a $O(mn)$ time complexity (Charras and Lecroq, 2004).

2.3.1.8. Skip Search Algorithm

The Skip Search Algorithm (Charras and Lecroq, 2004) is a string searching algorithm. It utilizes buckets of positions for each character of the alphabet. The preprocessing phase of the algorithm consists in computing the buckets for all the characters of the alphabet (Charras and Lecroq, 1997). The space and time complexity of this preprocessing phase is mirrored by the $O(m+\sigma)$ expression. One of the features of the Skip Search algorithm is quadratic worst case time complexity. It performs the expected number of text character inspections symbolized by $O(n)$ (Charras and Lecroq, 2004).

2.3.1.9. Knuth-Morris-Pratt Algorithm

The Knuth-Morris-Pratt Algorithm (Getoor, 2003) derives from the Morris-Pratt algorithm making comparisons from the left to the right. The preprocessing phase can be done in $O(m)$ space and time complexity and the searching phase can be done in $O(m+n)$ time (Charras and Lecroq, 1997). The maximum capacity of character comparisons during the searching phase is $2n$-1. The delay is bounded by $log_{\gamma}(m)$ where γ is the golden ratio (Charras and Lecroq, 2004).

2.3.2. Approximate String Matching Algorithms

The approximate string matching algorithm is focused on two operations: finding approximate substring matches inside a given string and finding dictionary strings that match the pattern approximately (Singla and Garg, 2012; Navarro and Baeza-Yates, 1999). It has many applications including text searching and pattern recognition often used in computational biology. These applications are beneficial in

many disciplines, from matching strings of DNA in genetics, to spell checking in a document and finding possible suggestions for misspelled words (Lopresto and Turner, 2000); from retrieving musical passages similar to a sample, to searching text under the presence of typing or spelling errors. Approximate string matching algorithms can be examined in 4 types, including edit distance based metrics, token based metrics, *q*-gram based metrics, and longest common subsequence based metrics.

Nico Jacobs, in his work (2004), reflects on definition of the term 'edit distance'. In his words: "The term 'edit distance' refers to the distance in which insertions and deletions have equal cost and replacements have twice the cost of an insertion". This sentence can be extended on the entire class of string metrics measuring distance as the number of operations necessary to transform one string into another. Several independent methods can be employed to define an edit distance depending on which of edit operations takes place: replace, delete, insert, transpose etc. Token based metrics are based on statistics of common words and are useful when word order is not important. The token-by token matching technique is more expensive than line-by-line matching since there are many tokens in a single line. In the following subsections basic approximate string matching algorithms are presented.

2.3.2.1. Levenshtein Distance

In 1965 Russian scientist Vladimir Levenshtein devised an algorithm called after his surname the Levenshtein distance or edit distance (Левенштейн**,** 1965). The distance, being a string metric, aimed at measuring the difference between two sequences. Edit distance between two words is defined as the minimum number of single-character edits, such as insertions, deletions or substitutions, required to convert one word into another. To refer specifically to Levenshtein distance a phrase edit distance is usually used. It is closely related to string alignments that occur in pairs. The greater the Levenshtein distance is, the greater the difference between the strings. Two units is the cost of simplest version substitutions with the exception to

the situation when the source and target are identical (Gupta *et al.*, 2009). At that time the cost is equal to zero. As for insertions and deletions the cost is one unit.

Upper left-hand corner of a two-dimensional array is the starting point for the algorithm. The array is indexed in rows by the letters of the source word, and in columns by the letters of the target (Martin, 2009). The algorithm gradually fulfils the array while finding all the distances between each initial prefix of the source and the target (Martin, 2009). Each cell described as [*i,j*] cell represents the minimal distance between the first '*i*' letters of the source word and the first '*j*' letters of the target word (Martin, 2009).

For example (Kumar and Chadrasekaran, 2011),

- If *a* is "rose" and *b* is "rose", then *lev*(*a,b*) = 0, because no transformations are needed. The strings are already identical.
- If *a* is "rose" and *b* is "hose", then *lev*(*a,b*) = 1, because one substitution (change "*r*" to "*h*") is sufficient to transform *r* into *h*.

The Levenshtein distance algorithm has been widely used in areas such as: spell checking, plagiarism detection, speech recognition, and DNA analysis (Turnbaugh, 2009).

The Levenshtein distance between two strings a and b is given by $lev_{a,b}(|a|,|b|)$ where

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{[ai \neq bj]} \end{cases} & otherwise \end{cases} \quad (2.1)$$

where $1_{[a_i \neq b_j]}$ is the indicator function equal to 1 when $[a_i \neq b_j]$, and 0 otherwise.

For similarity calculation the following formula is used:

$$sim(string1, string2) = 1 - \frac{lev((string1, string2)}{\max(|string1|, |string2|)} \quad (2.2)$$

2.3.2.2. Needleman-Wunsch

The Needleman-Wunsch algorithm was designed to align two nucleotide or amino acid sequences and find out their structural or functional similarity. Optimal sequence alignment is the target for a dynamic programming algorithm created by Needleman-Wunch (Needleman and Wunch, 1970). The algorithm is sometimes referred to as the Optimal Matching Algorithm. The idea behind Needleman-Wunch work has its roots in an observation that any partial sub-path that tends at a point along the true optimal path must itself be the optimal path leading up to that point (Musso, 2007). Hence, the optimal path can be found by incremental extension of the optimal sub-paths. In other words, a large problem can be divided into smaller problems. Solutions developed for small problems can be later combined together and applied to the large one. In a Needleman- Wunsch alignment, the optimal path must be stretched from beginning to end in both sequences (Musso, 2007). Taking it into consideration the term 'global alignment' has been coined. The Needleman-Wunsch algorithm takes quadratic time for a linear or affine gap penalty, so in general, it refers to a global alignment algorithm. In order to perform a Needleman-Wunsch alignment, a matrix is created which allows comparison of the two sequences (Musso, 2007). The score $M_{i,j}$ for every cell depends on the 3 cells corresponding to either or both sequence having 1 less letter (i.e. cells $M_{i\text{-}1,j}$, $M_{i,j\text{-}1}$ and $M_{i\text{-}1,j\text{-}1}$. It is calculated as follows:

$$\begin{aligned} M_{0,j} &= gap * j \\ M_{i,0} &= gap * i \\ M_{i,j} &= MAX\ (M_{i-1,j-1} + S(A_i, B_j),\quad M_{i\text{-}1,j} + gap,\quad M_{i,j\text{-}1} + gap) \end{aligned} \quad (2.3)$$

where *gap* is the gap penalty, *MAX* is the highest scoring cell, and the function *S* returns the score/penalty for matching the two corresponding letters. This algorithm performs alignments with a time complexity of O(*mn*) and a space complexity of O(*mn*). *m* is the length of the first string, and *n* is the length of the second string.

2.3.2.3. Smith-Waterman

The Smith–Waterman algorithm, first proposed by Temple F. Smith and Michael S. Waterman in 1981 (Smith and Waterman, 1981a), performs local sequence alignment to determine similar regions between two strings or nucleotide or protein sequences. It compares segments - rather than sequences - of all possible lengths optimizing similarity measures. The initial work of Needleman-Wunch has been modified after over a decade after its publication. The alteration facilitated local alignments. Reaching the boundaries of the search graph is not the obligatory condition for the alignment path as it can start and end internally. The score calculation, in which such facilitation was accomplished by adding '0' as a term, was described by Needleman-Wunsch. The Smith–Waterman algorithm, a dynamic programming algorithm (Smith and Waterman, 1981b) as with the Needleman–Wunsch algorithm, is a variation of the Needleman–Wunsch algorithm.

The Smith–Waterman algorithm requires O(*mn*) time to align two sequences of lengths *m* and *n*. For local alignments, the below equation is used.

$$\begin{aligned} M_{i,j} = MAX\ (&M_{i-1,j-1} + S(A_i, B_j), \\ &M_{i\text{-}1,j} + gap, \\ &M_{i,j\text{-}1} + gap, \\ &0) \end{aligned} \tag{2.4}$$

where *gap* is the gap penalty, *MAX* is the highest scoring cell, and the function *S* returns the score/penalty for matching the two corresponding letters (Rana, 2012).

2.3.2.4. Smith-Waterman Gotoh

The Smith-Waterman-Gotoh Algorithm is a distance-based dynamic programming algorithm for global pairwise alignment. The Smith-Waterman-Gotoh Algorithm (Gotoh, 1982) finds the minimum cost of aligning two sequences of lengths *m* and *n* in order *O*(*mn*) steps. The Smith-Waterman-Gotoh algorithm is designed to search for only one optimal (minimum-cost) alignment from among many others. Unfortunately, it does not always succeed.

2.3.2.5. Smith-Waterman Gotoh Affine Window

The Smith–Waterman Gotoh Affine Window Algorithm (Gotoh, 1982) is a similarity-based dynamic programming algorithm for local alignment with "affine" gap weights. The rule here is to affine a gap penalty. Two gap penalties, one for opening a gap and one for extending the gap (smaller than gap opening penalty) are utilized (May *et al.*, 2007). It fits better to the ideal gap penalty model than to the linear gap penalty model. Moreover, the requirement towards the algorithm is only to recognize if the previous alignment was to a gap or not. The dynamic programming application for the Smith–Waterman Gotoh Affine Window algorithm has time complexity of *O*(*mn*). This time complexity is similar to the algorithms employing linear gap penalty.

2.3.2.6. Hamming Distance

Named for Richard Hamming—who first introduced it in his fundamental paper on Hamming codes error detecting and error correcting codes in 1950 (Hamming, 1950) — the Hamming Distance between two strings of equal length is the number of positions at which the corresponding symbols are different (Belal *et al.*, 2012). It calculates the minimum number of substitutions required to turn one string into another. Alternatively, it computes the minimum number of errors that could have transformed one string into another (Herranz and Nin, 2009).

Telecommunication is one of areas where flipped bits in a fixed-length binary word can be counted thanks to the Hamming distance. As the distance's role is to estimate an error it is sometimes called the signal distance. Information and coding theories as well as cryptography are among disciplines where Hamming distance analysis of bits is utilized. Another application is to measure a genetic distance. It must be noted however that for comparing strings of different lengths or strings where apart from substitutions also insertions or deletions can occur, a more sophisticated metric should be provided. The example of such a metric is the Levenshtein Distance.

2.3.2.7. Jaro – Winkler

The Jaro–Winkler Distance (Winkler, 1990) is focused on measuring of similarity between two strings. The Jaro–Winkler distance is a variant of the Jaro distance metric (Jaro, 1989). Its task is to find records referring to the same entity in a data set within different sources (e.g. books, websites) (Jaro, 1995). When the Jaro–Winkler distance for two strings is high, then the strings can be considered similar (Winkler, 2006). The Jaro–Winkler distance metric is best suited for short strings such as peoples' names. The similarity score is normalized so that '0' indicates no similarity and '1' is an exact match (Top *et al.*, 2007).

m is the number of matching characters, t is half the number of transpositions, and p constant is a prefix scale which is usually 0.1. ℓ is the length of the common prefix shared by the two strings with a maximum of 3. d_w is the Jaro–Winkler distance and d_j is the Jaro distance of two given strings s_1 and s_2.

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m} \right) & \textit{otherwise} \end{cases} \tag{2.5}$$

$$d_w = d_j + \left(\ell p \left(1 - d_j \right) \right) \tag{2.6}$$

2.3.2.8. Q-Grams

Q-Grams substring relies on counting the number of the occurrences of different *q*-grams in two strings. The closer the strings are, the more *q*-grams in common they have (Ukkonen, 1992). It is primarily used in statistical natural language processing. According to the definition in Al Dummor (2010), "A *q*-gram is a subsequence of *q* items from a given sequence. These *q* items can be phonemes, syllables, letters, words, or base pairs." Monograms, bigrams, and trigrams are examples of a *q*-gram.

2.3.2.9. The Winnowing Algorithm

The Winnowing Algorithm (Schleimer *et al.*, 2003) is an efficient local fingerprinting algorithm used for detecting plagiarism. This algorithm uses hashing technique for document fingerprinting. Similarities between source code files can be effectively detected by the Winnowing Algorithm. The method relies on exploitation of a set of fingerprints composed of *k*-gram hashes. A fingerprint also carries positional information describing the document and the location within the document that the fingerprint origins from. If two documents share one or more fingerprints, it is extremely likely that they share a *k*-gram as well (Schleimer *et al.*, 2003). The Winnowing algorithm is very simple. First, whitespaces and variable names are removed and then "*n*-grams" are hashed. Each window stores only the smallest hash. Storage of the first hash in every window would add an extra character at the start of a file. In a consequence no matching fingerprints would be found for otherwise identical documents. As well as comparing fingerprints filenames, the program also compares size of the jar file, the number of entities in the jar file, and the name of the jar file itself. The minimum hash value should be selected in each window. If there is more than one hash with the minimum value, the rightmost occurrence must be selected. Then all selected hashes as the fingerprints of the source code are saved. The key property of the Winnowing Algorithm is that the selection of hash depends only on the contents of the window. It does not depend on any external information

referring to the position of the window in the file or on its connections with other windows. The advantage of the algorithm for hashing selection is the ease of detection the trade-off between the fingerprint length and the shortest matching string. A disadvantage of this method is that it gives no guarantee that matches between source codes are detected (Schleimer *et al.*, 2003). Moss uses the Winnowing Algorithm to detect plagiarism. Its time complexity is $O(n^2)$.

2.3.2.10. Block Distance

The Block Distance is the same as the Taxicab Geometry or Manhattan Distance. Taxicab geometry, explored by Minkowski in the 19th century, is a form of geometry in which the usual distance function or metric of Euclidean geometry is replaced by a new metric, named as taxicab. The taxicab distance between two points is the sum of the absolute differences of their Cartesian coordinates (Panigrahi, 2015). Rectilinear distance is another name of the taxicab metric, but not the only one. Interchangeably following terms are also used: L_1 distance or ℓ_1 norm, city block distance, Manhattan distance, or Manhattan length, with corresponding variations in the name of the geometry (Krause, 1987; Black, 2006). The name 'taxicab geometry' alludes to the grid layout of streets on the island of Manhattan. They cause the shortest path a car could take between two intersections in the borough to have length equal to the intersections' distance in taxicab geometry (Russell and Cohn, 2012). In this metric moving from one pixel to another is only possible alongside pixel grid lines. Diagonal moves are not allowed; hence term 'city block distance' is used. The `city block distance' given by the block distance d_1, between two vectors p, q in an n-dimensional real vector space with fixed Cartesian coordinate system, is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes. More formally,

$$d_1(p,q) = \| p - q \|_1 = \sum_{i=1}^{n} | p_i - q_i | \tag{2.7}$$

2.3.2.11. Cosine Similarity

In general, Cosine Similarity is a metric that determines similarity between two documents (Kalaivendhan and Sumathi, 2014). To be more precise, it determines the similarity between two vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° equal to 1, and for any other angle it is less then '1'. Two vectors with the same orientation have a Cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude (Giannakouris – Salalidis *et al.*, 2014). That's why it is a matter of orientation and not magnitude. Positive space is the environment where Cosine Similarity can be applied. Final results are within boundaries of '0' and '1'.

For text matching, the attribute vectors A and B are usually the term frequency vectors of the documents (Giannakouris – Salalidis *et al.*, 2014). Given two vectors of attributes, A and B, the cosine similarity, $\cos(\theta)$, is a normalized dot product of A and B as follows:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\| A \| \| B \|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2 \times \sum_{i=1}^{n} (B_i)^2}} \quad (2.8)$$

The resulting similarity ranges from 0 meaning exactly different, to 1 meaning exactly the same. 0 usually indicates independence, and remaining in-between values stand for intermediate similarity or dissimilarity (Giannakouris – Salalidis *et al.*, 2014).

2.3.2.12. Euclidean Distance

The "ordinary" distance between two points in Euclidean space is called Euclidean distance. One of the points could be measured with a ruler and

additionally is given by the Pythagorean formula (Deza and Deza, 2009). If the formula is understood as distance, Euclidean space (or even any inner product space) becomes a metric space (Ameela and Kaleeswaran, 2013). The Euclidean distance between point p and q is the length of the line segment being a connection between them $(\overline{pq})$ (Ameela and Kaleeswaran, 2013). In general, for an n-dimensional space, the distance is

$$d(p,q)=d(q,p)=\sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2}=\sqrt{\sum_{i=1}^{n}(q_i-p_i)^2} \quad (2.9)$$

2.3.2.13. Monge-Elkan

Monge and Elkan (1996) proposed a simple but effective method to measure the similarity between two text strings. The method relies on an internal character-based similarity measure and a token level similarity measure. It utilizes an internal similarity function $sim'(a, b)$ able to measure the similarity between two individual tokens a and b. Given two texts A, B, with $|A|$ and $|B|$ being their respective number of tokens, and an external inter-token similarity measure sim', the Monge-Elkan measure is computed as follows:

$$sim\,MongeElkan(A,B)=\frac{1}{|A|}\sum_{i=1}^{|A|}\max\{sim'(a_i,b_j)\}_{j=1}^{|B|} \quad (2.10)$$

2.3.2.14. Jaccard Distance

The Jaccard Distance is a proper metric. The Jaccard Distance, which measures dissimilarity between sample sets, is complementary to the Jaccard coefficient which measures similarity between sample sets, and is defined as the size

of the intersection divided by the size of the union of the sample sets (Stoilos *et. al.*, 2005). Given two strings *A* and *B*, the Jaccard Similarity is computed as follows:

$$similarity = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sum_{i=1}^{n} (A_i)^2 + \sum_{i=1}^{n} (B_i)^2 - \sum_{i=1}^{n} A_i \times B_i} \quad (2.11)$$

2.3.2.15. Overlap Coefficient

The Overlap Coefficient is a similarity measure related to the Jaccard index that computes the overlap between two sets defined as follows:

$$overlap(X,Y) = \frac{|X \cap Y|}{\min(|X|,|Y|)} \quad (2.12)$$

If set *X* is a subset of *Y* or the converse then the overlap coefficient is equal to one.

The Overlap Coefficient is a measure of similarity. The overlap proportion between two probability distributions is reflected by the Overlap coefficient. The coefficient is related to dissimilarity index and its properties. The maximum-likelihood estimation of the overlap between two normal distributions can be treated extensively. In such case the extensive treatment is presented as an example of estimating the overlapping coefficient from sample data (Takale And Nandgaonkar, 2010).

2.3.2.16. Dice Coefficient

The Dice's coefficient, independently developed by Sørensen (1948) and Dice (1945) as a statistical method, used to compare the similarity of two samples.

This method was published in 1948 by Sørensen and 1945 by Dice. Dice coefficient is used mainly for ecological community data (Looman And Campbell, 1960). In theory the use of Dice Coefficient can be justified as an intersection of two fuzzy sets. The reasons for its utilization however are more empirical than theoretical (Roberts, 1986). Given two strings *A* and *B*, the Dice Similarity is computed as follows:

$$similarity = \frac{2 \mid A \cdot B \mid}{\mid A \mid^2 + \mid B \mid^2} = \frac{2\sum_{i=1}^{n} A_i \times B_i}{\sum_{i=1}^{n} (A_i)^2 + \sum_{i=1}^{n} (B_i)^2} \tag{2.13}$$

2.3.2.17. Minkowski Distance

The Minkowski Distance is a metric applied on Euclidean space which is regarded as a generalization of both the Euclidean distance and the Manhattan distance (Dziuda, 2010). Whilst using Minkowski distance *p* is usually set at 1 or 2. The Minkowski Distance of order *p* between two points

$$P = \left(x_1, x_2, ..., x_n\right) \quad and \quad Q = \left(y_1, y_2, ..., y_n\right) \in \Re^n \tag{2.14}$$

is defined as (Madhulatha, 2011):

$$\left(\sum_{i=1}^{n} \left| x_i - y_i \right|^p \right)^{1/p} \tag{2.15}$$

2.3.2.18. Greedy String Tiling Algorithm

Greedy String Tiling Algorithm was introduced by Michael Wise (1993a). It is used to compare two token strings. When comparing two token strings, the aim is

to find a set of substrings that are the same. It identifies the longest plagiarized sequence of substrings from the text of the source document (the first textual document file) and returns the sequence as tiles from the source document and the suspicious document (the second textual document file). Any token of the first textual document file may only be matched with exactly one token from the second textual document file. Substrings are to be found independent of their string position, but the relative positions do not change. As long substring matches are more reliable than short ones, due to their better accuracy, they are utilized more often. It has worst case complexity O(n^3), where n is the sum of the lengths of the input strings. The detailed information about the Greedy String Tiling algorithm is described in Section 3.

2.3.2.19. Heckel's Algorithm

Heckel's Algorithm (Heckel, 1978) is an algorithm to detect source code similarity. In order to compare two files, it takes a line as the basic unit, although word, sentence, paragraph, or character units are possible. Similarities are found until only differences remain. Two observations are made. The first observation is that a line that occurs only once in each file must be the same line. The second observation is that if in each file adjacent to a found line pair there are lines identical to each other, these lines must be the same line. YAP2, one of the YAP Programs, uses Heckel's Algorithm to detect plagiarism (Wise, 1996). Its overall complexity is $O(mn)$.

2.3.2.20. Running Karp-Rabin Greedy String Tiling Algorithm

Tiling one string with matching substrings of a second string is the operational method of Running Karp-Rabin Greedy String Tiling (RKR-GST). RKR-GST (Wise, 1995) is able to detect transposed substrings. Karp and Rabin's Algorithm, designed for fast substring matching, is arguably the earliest version of fingerprinting based on k-grams. Substrings of some predefined fixed length are the

subject of the search performed by the RKR-GST algorithm. Hash value for the entire pattern string is not stable. A new hash value is created for each unmarked substring of the pattern string length for the substring. In the next step, each of the hash values for the pattern string is compared against the hash-values for text string. After each iteration, the length of the matches is decreased until the minimum match length is reached. The worst-case complexity of Running Karp-Rabin Greedy String Tiling is $O(n^3)$. YAP3, one of the YAP (Yet Another Plague) plagiarism detection programs, uses Running Karp-Rabin Greedy String Tiling Algorithm to detect plagiarism.

2.3.2.21. Longest Common Subsequence

The Longest Common Subsequence (LCS) Algorithm (Zeidman, 2010) is used to find the subsequence of maximal length that appears in all sequences in a set of just two sequences of items. It is based on the concept of creating fingerprints from hashes of *k*-grams guaranteed to detect copies. A *k*-gram is a continuous substring of numbers or letters of length *k* created by dividing the document into *k*-grams where *k* is a parameter chosen by the user. This parameter, a simple formal statement of the problem, finds the longest common subsequence with gaps. So it yields good results. However, LCS has some disadvantages, but the main disadvantage concerns the computational problems. In the worst case, it can take time O(*mn*) (where m is the sum of tokens and n is the sum of the lengths of the input strings). The LCS's only implementation takes in worst-case O(*mnlogn*) time and O(*mn*) space for file comparison.

2.4. Stemming and Thesaurus

Lovins (1968) defines a stemming algorithm as "a procedure to reduce all words with the same stem to a common form, usually by stripping each word of its derivational and inflectional suffixes." Ekmekçioğlu (2003) defines "Stemming is generally effected by means of suffix dictionaries that contain list of possible word

endings, and this approach has been applied successfully to many different languages".

There is an abundance of software that can be used to find the roots of Turkish words. The most popular ones are Zemberek[21], Snowball (Porter, 2001), Preto (Tunalı and Bilgin, 2012), the first 5 characters stemming (Can *et al.*, 2008), TRMorph (Çöltekin, 2010), and PCKimmo (Koskenniemi, 1983).

Zemberek is a software that helps to find the roots of Turkish words (Akın and Akın, 2007). It is a project oriented to the Turkish language that refers to topics such as Natural Language Processing or spell-checking for Turkish. It is a free-software, surface-form oriented, morphological analyzer, which fills an important gap. It is an open-source Java library and is used as a syntax checker for Turkish and Turkic languages in many software packages. Zemberek gives all possible results for the morphological analysis of a word. For the words that may be mistyped, Zemberek provides a function for checking the word and also, if it is not a correct word, a function for suggesting a different word. Zemberek converts a Turkish text without Turkish letters into a text with proper Turkish letters for words that may be written from a non-Turkish keyboardIt provides the root forms of a given word using NLP algorithms and a dictionary-based root parser (Yılmaz, 2009). It is successfully used not only in academic research, but also in real-world applications, such as openoffice.org[22] and Turkish Pardus Linux distribution[23].

The application of affix stemmer was performed by Evren (Kapusuz) Çilden (2006) with Snowball2 software. Snowball is a string processing language program that is specifically designed for stemming purposes[24]. Snowball is a free, small character processing language program designed to create algorithms to find roots. The algorithms for finding roots have been developed for many languages using Snowball (Yüksel *et al.*, 2010; Porter, 1980).

[21] http://code.google.com/p/zemberek/
[22] http://www.openoffice.org.tr
[23] http://pardus.org.tr
[24] http://snowball.tartarus.org/

The first 5 Characters Stemming uses only the first 5 characters in all words (Can *et al.*, 2008; Yılmaz *et al.*, 2009). Preto[25] is a powerful and measurable pre-process tool which is specially developed with cross-platform, stemming, stopword filtering, statistical terms filtering and *n*-gram creation with a wide range of pre-processing options for Turkish text pre-process (Tunalı and Bilgin, 2012).

TRMorph (Çöltekin, 2010) is a freely available, two-level morphological analyzer for Turkish based on Stuttgart Finite State Transducer tools. It is capable of the resulting part-of-speech (POS) tags after each derivation.

PCKIMMO, developed by Antworth (1990) at the Summer Institute of Linguistics (SIL), is a new implementation for microcomputers of the KIMMO (named after its inventor Kimmo Koskenniemi) program (Koskenniemi, 1983). Computational linguists, descriptive linguists, and those developing natural language processing systems are the group of people interested in the program. It is the most well-known version of Koskenniemi's two-level processor.

The postulate of a morphological analysis in the Turkish stemming algorithms was raised by Sever and Bitirim (2003). In their study they evaluated the effectiveness of FINDSTEM – a new stemming algorithm operating on Turkish documents and queries. In addition, they compared the functionality of the algorithm with two other, previously developed, Turkish stemmers: 'A-F' and 'L-M' algorithms. 'A-F', similar to FINDSTEM, relies on inflection and derivation while 'L-M' is based solely on inflection. The results of the comparison indicated FINDSTEM superiority over remaining two algorithms. FINDSTEM obtained higher precision values on the scale of 11-point recall levels in respect to document indexing and search.

Bilgin *et. al.* (2004a) claims that WordNet is a lexical data base grouping words into sets of synonyms initially created for the English language. As a part of so-called Balkanet project a WordNet for Turkish has been initialized. Two applications implementing the Turkish vocabulary had been summarized and links to the resources passed to the WordNet builders. The first application (Bilgin *et. al.*, 2004b) is an attempt to export semantic relations to other WordNets. It is based on

[25] http://code.google.com/p/preto/

purely morphological processes in the exporting language. The second application (El-Kahlout and Oflazer, 2004) is a WordNet-based "reverse dictionary" which delivers words that correspond to a definition given by users. Basic statistics on the current status of Turkish WordNet (March 2004) are examined in this study. According to these statistics, there are 11,628 synsets, 16,095 synset members, 987 unlexicalized concepts, 4,913 definitions and the average synset size is 1.38.

In the study by Amasyalı (2005), four methods were proposed for automatic generation of some main parts of Turkish Wordnet. Two methods were implemented and hyperonym relations are generated automatically with 66% success ratio. The methods are not only for a special relation, but they can also be implemented for other relations.

Birant's study (2008) aims to prepare the structure of a language as a module that works in high performance by not making concessions in its correctness in order to make this module work with past and future projects. The study of Aktaş (2006) is accepted as the base in this study. The ratio of the correctness was approximately 99% in the study. Some structures were standardized by making some revisions. Standardization works were done by using the XML structures which were used to send information between the modules, and as the result of the program that was run for any morphological work and the source of the rules to separate the suffixes correctly. The suffixes are allowed to be tagged as readable and efficient for performance by being abbreviated and systematically named according to predefined rules.

In the study by Amasyalı and Beken (2009), words were placed in a semantic space. The word's coordinates in semantic spaces need the similarity of the words according to their meanings. Harris (1968) stated that two words' semantic similarity is related to the number of documents in which both words are found. They used his hypothesis for Turkish words. Firstly, they obtained a word co-occurrence matrix from a web corpus. Then, the numerical coordinates of the words were calculated by using multidimensional scaling. Texts coordinates were obtained from word coordinates, which passed in the texts. In their experiments, Turkish news texts were classified into five classes. Their approach, which is not for only Turkish

words/texts, but also for all other languages, received more successful results than the traditional Bag of Words space.

2.5. Web Mining and Data Mining for LMS

As the name clearly suggests Web mining is the application searching through web data where it discovers potentially useful and previously unknown information or knowledge (Ajoudanian and Jazi, 2009). The application of Web mining varies from relevant information capturing and new knowledge creation out of the relevant data to personalization of information and learning about consumers or individual users. Web mining, a part of the Information Retrieval and Information Extraction systems, helps to understand customer's behavior as well as evaluate the performance of a web site what indirectly helps to boost a business (Ajoudanian and Jazi, 2009).

Web mining can be divided into three different types (Liu, 2011):

1. Web Usage Mining,
2. Web Content Mining, and
3. Web Structure Mining.

Web Content Mining is the extraction of useful information, which is relevance to the suggested query, from content such as text, image, audio, video, hyperlinks, metadata, and extracts work performed by search engines (Kosla and Blockeel, 2000; Inamdar and Shinde, 2010). Web Content Mining examines the content of the web as well as the result of the search (Ajoudanian and Jazi, 2009).

Web Structure Mining which focuses on the hyperlink structure of the Web, aims at extracting previously unknown interconnections between Web pages (Dunham, 2003). Web mining links different objects in an untraditional way avoiding a trap of jumping to wrong conclusions. The appropriate handle of the links facilitates potential correlations selection and improve the predictive accuracy of the learned models (Getoor, 2003).

Web Usage Mining primarily deals with understanding user behavior in interacting with the Web or with a website (Gupta, 2014). The above refers to the automatic analysis of patterns in a click stream and associated data collection being the result of user interactions with Web resources on one or more Web sites (Liu, 2011; Cooley, 1997; Mobasher, 2006; Srivastava, 2000). Capturing, modeling, and analyzing of behavioral patterns and profiles of Web site users are the paramount goals of Web Usage Mining (Liu, 2011). The mined data often includes data logs of users' interactions with the Web (Gupta, 2014).

There are applications of web usage mining for learning management systems. In the following paragraphs these applications are summarized.

In the study by El-Halees (2009), a case study was given in educational data mining to analyze learning behavior of students in the Moodle System of a Database Course at Islamic University of Gaza in the first semester of 2007-2008. After collecting data from 151 students, preprocessing them, and discretizing the numerical attributes to categorical attributes, the data mining techniques — which are outlier detections, classification, clustering and association rule mining — were applied using Knime data mining tool. The rules obtained from association rule mining were sorted by lift metric. According to the discovered rules, if the attendance of the student is good, the student does e-exercises, and the student has good midterm grade, the rule has the highest lift. It means that this rule has highest correlation between attendance and final grade. J48 decision tree was used for classification technique to represent logical rules of students' final grade. The reason for using J48 decision tree was to predict student failures by low grades before the end of the course semester. The Expectation-Maximization Algorithm (EM-clustering), a clustering method, was used to group students according to their behavior and performance. Using the results of EM-clustering algorithm students were divided into five groups. In this study, nine attributes were used, including attendance, gpa, hours, e-resources, e-exercise, e-homeworks, midterm, lab, final and class grades. After the association rule mining, classification and clustering processes, the outlier detection was used to detect outliers in the student data.

Romero *et. al.* (2013) studied the performance of different methods of the data mining for classifying students by using seven different datasets from engineering courses at Cordoba University. They developed a specific educational data mining tool integrated into the Moodle system for predicting final grades of the students that use courses in the Moodle. This tool was implemented in Java using the KEEL framework. Moodle data had to be preprocessed to convert it into a text file in KEEL format. Three dataset were used in the study. The first dataset was from 138 students, the second dataset was from 135 students, and the third dataset was from 438 students. The numerical data was discretized using the manual method and the equal-width method. 10-fold cross validation method was used. After the preprocessing phase, classification algorithms were used. For the data mining phase, nine attributes were used including course id, number of assignments completed, number of quizzes passed, number of quizzes failed, number of messages sent to the forum, number of messages read on the forum, total time used on assignments, total time used on quizzes, total time used on forum, and final grade that the student obtained in the course. Two experiments were carried out to evaluate the performance of several classification algorithms to predict the final grades of the university students based on the Moodle logs. For statistical classifier, ADLinear, PolQuadraticLMS, Kernel, and KNN algorithms were used. For classification, decision tree, C4.5 and CART algorithms were used. For rule induction, AprioriC, CN2, Corcoran, XCS, GGP, and SIA algorithms were used. For classification using fuzzy rule learning, MaxLogitBoost, SAP, GAP, GP, and Chi were used. For classification by neural networks, NNEP, RBFN, GANN, and MLPerceptron were used. The best algorithms for the first dataset were CART, GAP, GGP, and NNEP with the about 65% accuracy. The best algorithms for the second dataset were CART and C4.5 with the about 65% accuracy. The best algorithms for the third dataset were Corcoran, XCS, AprioriC, and MaxLogicBoost with the about 60% accuracy. According the test results, it is very difficult to predict the final grades of the students based on web usage data because the accuracy obtained is in the range of 65%.

In the study by Dimić *et. al.* (2010), an investigation of data mining methods for Moodle courses was presented. For the purpose of their study, they collected and

analyzed data on 130 students in the Programmable Logic Devices course during the 2008-2009 school year at the Electrical Engineering and Computing Science College in Belgrade. For the purpose of their research, they collected data on students in the course. They first collected information on students' access to the Moodle course. For data mining phases, they used Weka tool. Five attributes were used, including number of read lessons, number of solved interactive tasks, number of used PPT presentations, number of taken self-check tests for the exam preparation, and students' final grade. They applied the equal-width discretization method and the manual discretization method. The range of the attribute numerical values were divided into three intervals—low, medium, and high—with the method equal-width and grouped into five groups—failed, passed, good, very good, and excellent— with the manual discretization method. The aim of this study was to find clusters of students with similar learning characteristics, group active students with inactive students in order to improve activeness and learning characteristics of inactive students. They used the Expectation-Maximization Algorithm (EM-clustering) for clustering students into five clusters according to their activities in the course, their various learning materials and their grade obtained in the exam. Also, the J48 decision tree method was used to predict the final grades of students. According to the obtained rules from classification, learning material and the final exam have a connection. When a student used self-check tests, read lessons, solved interactive test for an unlimited number of times, then he/she passed the exam in 85% of cases with good, very good or excellent.

In the study by Casey and Gibson (2010), the statistics of Moodle daily course views from an Irish third-level institution were analyzed. The analysis data collection took place between September 2009 and May 2010 during a number of modules in Computer Science and MSc Digital Media courses for the first-year and final-year BSc students. 250 students took the BSc I course, 50 students took the BSc IV course, and 89 students took the MSc DM course. The correlations between the statistics and students' results on degree and masters courses were examined. The other aim of this study was to analyze student usage patterns to use Moodle

effectively. According to analysis results, some associations between student behavior on Moodle and their final grade were found.

In the study by Pedraza-Perez *et al.* (2011), a Java desktop data mining tool for Moodle logs was proposed. The application was incorporated into Java computing platform within KEEL framework. It provides a wizard that has a simple interface. It is based on options to facilitate the execution of the data mining process. For preprocess phase, the numerical values were divided into the categorical values, the final grade was added to the data, and the data was split into training and test data files. For data mining phase, classification, regression, and association rule mining algorithms were provided.

In the study by Jailia and Tyagi (2013), a use of data mining techniques — especially classification technique in Moodle using RapidMiner data mining tool — was made to predict students' performance, to help teachers to improve learning, and to understand their students' learning. The aim of this study was to help discover pedagogically relevant knowledge from Moodle using data mining techniques. After the data was cleaned, transformed and preprocessed, the decision tree algorithm, a classification algorithm, was applied to evaluate and predict the students' performance. In this study, 13 attributes were used for the data mining process, including user information, student information, logs of every user's action, assignment information, submitted assignment information, chatroom information, glossary information, forum information, number of posts to the forums, quiz information, number of quiz attempts, the final quiz grade, and class performance. According to the classification results, the instructors can predict the performance of the students using a data mining tool in Moodle.

In the study by Mohd (2013), the application GenARM was developed in the implementation of Association Rule Mining in grouping Moodle data. The aim of this study was to analyze and group the learning behavior of 30 students who visited Moodle course on the BCS3203-Web Scripting using the Association Rule Mining. For data mining phase, twelve attributes were used, including course id, login information of the user (username), login time, time of action, action and action information, number of quiz views, number of assignment views, number of forum

views, number of page views, number of resource views, number of URL views, and number of course views. To generate the association rules, the data was transformed into binary values. Visual Basic.Net is used as a platform to develop GenARM. According to the clustering results, assignment views, course views and page views were more popular modules than quiz views among the students. This study and the GenARM application proved that the association rule mining is suitable in data mining from Moodle and clustering student learning behavior can be completed using association rule mining. This application is able to generate strong rules.

In the study by Bovo *et. al.* (2013), an application which uses data mining and machine learning methods related to clustering and classification was proposed. The aim was to solve the problem of student monitoring in e-learning. This application was used not only in Moodle, but also used in another LMS; however, in this study, Moodle was selected to mine the data. All the clustering experiments which were used with Expectation Maximisation, Hierarchical Clustering, Simple K-Means and W-Means with using 3 as parameter were performed using Weka to analyze student groups. To test the accuracy of the clusters, the 10-fold cross-validation method was used. The test showed that the X-Means clustering algorithm was the best algorithm among the used algorithm in the test. According to the test results, 2 or 3 clusters were sufficient to monitor and identify students who were about to failure.

Data mining has been gaining on importance in the recent years, being utilized in science and business to analyze large amount of data, unveiling various trends that otherwise would not be discovered. Different data mining application rely on different mining techniques. In this regard someone can distinguish association rules mining, sequential patterns mining, classification, and clustering (Chen *et al.*, 1996).

The study of Romero *et. al.* (2008) was a survey of data mining application in the Moodle system. Researchers used Weka and Keel systems to apply data mining. The data was from 438 students of 7 courses. The numerical data was divided into the categorical data using the manual method and the equal-width method. For the data mining phase, 12 attributes were used including course id, number of assignments completed, number of quizzes completed, number of quizzes

passed, number of quizzes failed, number of messages sent to the teacher, number of messages sent to the chat, number of messages sent to the forum, number of messages read on the forum, total time spent in assignments, total time used on quizzes, total time used on forum, and the final grade the student obtained in the course. Romero *et al.* (2008), in their study, observed a group of students from a specific course that used Moodle learning platform in their education process. Basing on the activity of the learners on the platform and the number of tasks they completed on Moodle, Romero *et al.* (2008) distinguished three groups of students: very active, active, and non-active. Also the other aim of Romero *et al.* (2008) was to classify the students into three groups — fail, excellent, and good — with equal final grades depending on the activities carried out in Moodle. They grouped the students using the K-Means Algorithm, with a value of 3 to the number of clusters. Romero *et. al.* (2008) classified the students using the C4.5 Algorithm with 3-folds cross validation. They used the Apriori Algorithm (Agrawal *et al.*, 1993; Agrawal and Srikant, 1994) for finding association rules with a minimum support of 0.3 and a minimum confidence of 0.9 as parameters with using Weka. They have developed a specific Moodle data mining tool to preprocess or apply data mining techniques. This tool was integrated into Moodle to carry out all data mining processing. According to the rules obtained, the number of messages read in the forum, the number of messages sent in the forum, the number of passed quizzes, and the total time were significant attributes to determine the final mark of the students.

In the study by Romero *et. al.* (2009), the application of subgroup discovery using evolutionary algorithms to the student's usage data of the Moodle at the University of Cordoba was described. The data was from 293 students in 5 courses. In addition, Romero *et al.* (2009) focused on discovering rules determining the students' usage of Moodle and relations between level of activity on the platform and achieved grades, and to show the discovered rules and their measures to the instructor. The results obtained by Apriori-SD, CN2-SD, and CBA Algorithms, deterministic algorithms for subgroup discovery, were compared with the SDIGA algorithm, a nondeterministic algorithm, and thanks to the evolutionary model of the extraction of fuzzy rules a small number of rules that are easy to comprehend by

instructors have been selected. The subgroup discovery algorithm was implemented using the Keel tool. For the data mining phase, 11 attributes were used, including course id, number of assignments completed, number of quizzes completed, number of assignments passed, number of quizzes passed, number of assignments failed, number of quizzes failed, number of messages sent to the teacher, number of messages sent to the chat, number of messages sent to the forum, number of forum messages reads, and one class attribute which was discretized student's grade. The finding rules showed that quizzes are directly related to the received grades. Furthermore, students not dedicating time to assignments, forum participation and quizzes have difficulties in passing an exam.

In the study by Romero *et. al.* (2010), the use of Rare Association Rule Mining (RARM) over educational data of students from Moodle courses on computer science at University of Córdoba was studied. The data was from 230 students in 5 courses. Several Association Rule Mining (ARM) and RARM algorithms were compared with using the Moodle course data in this study to discover rare class association rules from the data. After the data was discretized, the data mining process was begun. For the data mining phase, 11 attributes were used, including identification number of the course, number of assignments completed, number of quizzes taken, number of quizzes passed, number of quizzes failed, number of messages sent to the forum, number of messages read on the forum, total time spent on assignments, total time spent on quizzes, total time spent on forum, and students' final grade. For Apriori-Frequent, the minimum support value was taken as 0.05. For Apriori-Infrequent, Apriori-Inverse, and Apriori-Rare, the maximum support value was taken as 0.1. The results of the rules extracted using Apriori-Frequent algorithm showed that quizzes were related to the final grade. According to the results of comparison of ARM and RARM algorithms, the confidence values of rare association rules were higher than the confidence values of frequent association rules, and the support values of rare association rules were lower than the support values of frequent association rules. The results of these rules found using association rule mining and rare association rule mining algorithms showed that the

rare association rules had high confidence levels, and these algorithms can help the instructor to detect infrequent student behavior in Moodle.

In the study by García *et. al.* (2010), some of the main drawbacks of association rule algorithms to data generated by e-learning system were described. They used Weka over Moodle logs to apply the association rule mining in LMS. The aim of this study was to discover information for making decisions about the students' usage data of the Moodle activities in order to improve the students' learning. For data mining phases, they used mdl_assignment, mdl_chat, mdl_forum, mdl_quiz, and mdl_log tables from Moodle. Twelve attributes were used, including course id, number of realized assignments, number of realized quizzes, number of passed quizzes, number of failed quizzes, number of messages sent to the chat, number of messages sent to the teacher, number of messages sent to the forum, number of messages read on the forum, total time used in assignment, total time used in quiz, total time used in forum, and the students' final grade. They used the manual discretization method for the grade attribute, which are fail, pass, good and excellent, and the equal-width discretization method for the other eleven attributes, which are low, medium and high. García *et al.* (2010) applied the Apriori algorithm to find association rules within the discretized data with a minimum support of 0.3 and a minimum confidence of 0.9 as parameters. Among the obtained association rules, while some rules were uninteresting, some rules were useful for the instructor. So by means of the useful rules obtained, the instructor can detect the student with learning problems and predict the final grade of the student.

In the study by Lile (2011), an educational data mining model for Moodle data was proposed. This model was based on the attribute selection, clustering, classification, and association rule mining techniques. For attribute selection Information Gain, Relief, Chi-Squared, and Uncertainty were used. For clustering K-Means, for classification Tree Induction, for association rule mining Apriori, FPGrowth, and Generalized Sequential Patterns (GSP) were used. The aims of this study were to discover useful knowledge from Moodle, to analyze the learning processes by applying data mining techniques, to improve the learning experience of students, to group students to investigate their behavior, to define variables which

affect the final grades, and to discover relationships between variables. According to the experimental results, the data mining model presented in this study was able to obtain comprehensible and logical feedback from the Moodle data describing students' learning behavior patterns. RapidMiner and Weka tools were used to mine data of 29 students from the "C Programming -CEN112," which was the Moodle course used at Epoka University in the 2009-2010 semester. The main activities of Moodle are classified into: assignment view, resource view, archive view, virtual classroom participation, forum read, discussion read, forum post, discussion, and post assignment upload. For the data mining phase, 11 attributes were used, including name of user, number of courseware and other supporting materials views, number of chat messages, number of blog views, number of forum reads, number of forum posts, number of discussion reads, number of assignments views participations, number of assignment answer uploads, quiz result score, assignment result score, and final grade. According to the discovered rules with the minimum support value as 0.3 and minimum confidence value as 0.8, the assignment score and the quiz score served as predictors for the final grade.

In the study by Kularbphettong and Tongsiri (2012), the data mining techniques showed that Moodle courses could be useful to model the students' motivation behavior. This study aimed to analyze student motivation on Moodle courses based on data mining techniques using Weka tool. The data for the analysis which was 3140 records was collected from the Information Technology for Communication and Learning Moodle Course at Suan Sunandha Rajabhat University during the 2010-2011 semester. After the data was collected, it was discretized with the equal-width method and preprocessed. Nineteen attributes were used, including identification number of pretests, identification number of posttests, grades obtained from pretests, grades obtained from posttests, time spent on pretests, time spent on posttests, total grades obtained from all pretests, total grades obtained from all posttests, total time spent on viewing materials, total time spent on uploading materials, attendance in class, identification number of assignments, grades obtained from assignments, class test, grades obtained from class tests, project scores, midterm scores, final scores, and class grade. The Apriori Algorithm was used in this

study with a minimum support of 0.2 and a minimum confidence of 0.9 as parameters to show the relationships between the Moodle activities and the final grade. The data was split into the train and test data. While 70% of the data was used for test, 30% of the data was used for training. According to the classification results, the accuracy of ID3 algorithm was 83.9161%, and the accuracy of J48 algorithm was 68.5315%. The results showed that these data mining methods can be useful to determine the outcomes of the student behavior in Moodle courses.

In the study by Chajri *et. al.* (2014), an educational data mining model for Moodle logs was proposed based on Information Gain, Decision Tree, K-Means, and FP-Growth to discover useful knowledge from Moodle data. They used Apriori and FP-Growth Algorithms for determining frequent itemsets. The aim of this study was to obtain the exact groups of students and to classify students. The data was from the 30 students of 4 courses. For the preprocessing phase, the data was discretized using equal-width method and the numerical data was divided into categorical data. For the data mining phases, they used RapidMiner tool. They used the mdl_assign, mdl_assign_submission, mdl_course, mdl_quiz, mdl_quiz_attempts, mdl_quiz_grade, mdl_forum, mdl_forum_post, mdl_forum_discussion, mdl_forum_read, mdl_grade_grades and mdl_log tables from Moodle. Eleven attributes were used, including course id, number of assignments completed, number of quizzes completed, number of quizzes passed, number of quizzes failed, number of posts in the forum, number of reads in the forum, time spent on assignments, total time spent on forums, total number of course materials and resources view, and final grade. The results showed that the total accuracy was 85.67%. According to the results, these rules can help to classify students.

3. MATERIAL AND METHOD

3. 1. Material

Moodle, which is open-source software, was used as a distance education tool for our thesis; Zemberek was used to stem words in texts, and the Greedy String Tiling Algorithm was used during the design of our plagiarism detection algorithm. Our proposed algorithm was compared with SIM, CopyCatch Gold, WCopyfind, Sherlock, Text Matching System and Document Similarity developed by YTÜ Kemik Group, Ferret and Turnitin because these systems are the most widely used plagiarism detection softwares.

3.1.1. Moodle

This study was targeted on making a plugin to Moodle Software that detects plagiarism of textual assignments written in Turkish. As assignments are important for education in any field, in this study, our aim was to increase the efficiency of grading textual assignments. Moodle was selected as a distance education tool for this study because it is one of the most widely used open-source Web Based Distance Education Models, and it is free and user-friendly. It also includes all features of an open-source learning management system and has features to improve educational quality. Although Moodle is used for distance education in our study we use it to support formal education. Our department uses the Moodle system to support our formal education. The detailed information about Moodle is given in Section 2.1.2.

3.1.2. Turkish Dataset

In this thesis, assignments of the Çukurova University Computer Engineering Department students primarily uploaded to Moodle system[1] were used for the experimental evaluation of the proposed plagiarism detection system. This dataset

[1] bmb.cu.edu.tr/moodle/bmb/moodle/

was created from the Data Organization and Management Courses of our Moodle web site. There are five assignments in this course and in each assignment the submitted assignment number is different. In the first assignment 37 students submit their assignment. In the second, third, fourth and fifth assignments, the submitted assignment numbers are 38, 32, 29, and 23, respectively.

In addition to assignments uploaded by the students, we also created some small datasets to compare our proposed system with the existing tools. These datasets are explained in the 4th section.

3.1.3. Stemming

Turkish, an agglutinative language, belongs to the Ural-Altaic language family. The meaning of words can be quite different when compared to other languages, such as English. Words occur with adding the affixes and suffixes to the word roots. Each morpheme has different information. Thousands of new words can be derived from morphemes by adding to the root words. Several rules cause morphemes in different words to take on various forms. Therefore, the access of word stems is important for language studies (El-Kahlout and Oflazer, 2006). In this study, we needed to find the roots of words to develop an effective system.

In this study, we tested a small Turkish data set with Zemberek. The results were more successful than the affix stripping method and the Snowball software. This is why we preferred Zemberek for stemming in this study.

3.1.4. The Dictionary of Synonyms

A dictionary is a collection of words in a specific language laid down in accordance with a given rule. The words are often listed alphabetically in dictionaries. There are definitions, etymologies, phonetics, pronunciations, and other information of the words in the dictionaries (Agnes, 1999; Aktaş *et al.*, 2013). Different words of the same meaning are called synonyms. Aktaş *et al.* (2013) suggested that “Synonyms may have identical or very similar meanings, which are

used to communicate easily by expressing thoughts appropriately". In addition, they noticed advantages of such dictionaries also reflected in the works of other scholars (Aktaş *et al.*, 2013). The application of the dictionaries can vary from automatic data extraction from a large text corpus, to query clustering for the support of search engines which use a question-answer structure (Wen et al., 2002). They are also used to index procedures, and to assign each word-stem to a concept class automatically (Salton, 1971). Additionally, the dictionaries automatically help to identify the author through a lexical choice (Reiter and Somayajulu, 2002). Verbs' conceptual structures and event types can be described with synonyms. In the latter case one can talk about delivering more complete verb frames for syntactic parser software (Huang and Hong, 2005). Another possible function of dictionaries is production of more lexically cohesive texts for authors from various fields (Donnely, 1994). Eventually, they can unveil the interactional relationship between syntax and semantics (Chief, 2000).

The Turkish WordNet was investigated because of the need of the Turkish Synonyms dictionary in the scope of the study. WordNet groups the English words into peer groups called synset. It provides a brief and general definition and saves the various semantic relations between these synonym groups (Zhang and Heflin, 2010). WordNet does not yet include the Turkish language. Under the project of BalkaNet[2], a Wordnet work has started for Turkish, but because the manual development process is difficult and time consuming, Turkish WordNet is not efficiently as the English language (Amasyalı, 2005). Since a Wordnet for Turkish has not been created yet, we prepare our own synonyms dictionary in this study. For this purpose, we used printed synonyms dictionaries prepared for Turkish Language. Our synonyms dictionary was prepared from the The Turkish Language Association (TDK) thesaurus, and dictionary of Kandemir and Kandemir (2012). The synonyms are for 5595 words in our dictionary. Table 3.1 shows a small portion of our synonyms dictionary that was used in this thesis.

[2] http://www.dblab.upatras.gr/balkanet/

Table 3.1. A Small Portion of Our Synonyms Dictionary

Word	**Synonyms**			
abartı	abartı	mübalağa		
abartılı	abartılı	mübalağalı		
abartmak	abartmak	büyütmek	mübalağa_etmek	şişirmek
bağ	bağ	demet	denk	deste
bağdaşmak	anlaşmak	bağdaşmak	uyuşmak	uzlaşmak
çaba	ceht	çaba	emek	gayret
çabalamak	ceht_etmek	çabalamak	çalışmak	gayret_etmek
çabuk	acele	çabuk	ivedi	tez
çabuklaşmak	çabuklaşmak	ivedileşmek	süratlenmek	tezleşmek
çağdaş	asri	çağdaş	modern	muasır
dağılmak	çözülmek	dağılmak	saçılmak	yayılmak
dahili	dahili	iç		
…	…	…	…	…

In this thesis, we also used the stemmed version of the synonyms list. For stemming, we used Zemberek and the stemmed version of the list in Table 3.1 is given in Table 3.2.

Table 3.2. A Small Portion of Our Stemmed Synonyms Dictionary

Word	**Stemmed Word**	**Stemmed Synonyms**			
abartı	abart	abart	mübalağa		
abartılı	abartı	abartı	mübalağalı		
abartmak	abart	abart	büyü	mübalağa_etmek	şişir
bağ	bağ	bağ	demet	denk	deste
bağdaşmak	bağdaş	anlaş	bağdaş	uyuş	uzlaş
çaba	çaba	cehit	çaba	emek	gayret
çabalamak	çabala	ceht_etmek	çabala	çalış	gayret_etmek
çabuk	çabuk	acele	çabuk	ivedi	tez
çabuklaşmak	çabuk	çabuk	ivedi	sürat	tezle
çağdaş	çağdaş	asri	çağdaş	modern	muasır
dağılmak	dağıl	çöz	dağıl	saç	yayıl
dahili	dahili	dahili	iç		
…		…	…	…	…

3.1.5. The Abbreviation List

There are many abbreviations in Turkish that are used to save space. Especially, in information systems, students use abbreviations to enable plagiarism. This is why we created our abbreviation list to check homeworks submitted in computer science courses. Since most of the computer terms are used as English, all abbreviations written in Turkish and English about computer science and information science are transformed to Turkish phrases. Our abbreviation list was prepared from the Informatics Association of Turkey's (TBD) provision glossary of information terms, the abbreviation dictionary of TDK, and Wikipedia. There are 232 abbreviations in our list. A small portion of the list prepared for the Turkish Language is presented in Table 3.3.

Table 3.3. A Small Portion of Our Abbreviation List

Abbreviation	**Word**
ai	yapay_zeka
al	assembly_dili
alu	aritmetik_mantık_birimi
amb	aritmetik_mantık_birimi
amo	arama_motoru_optimizasyonu
ansi	amerikan_ulusal_standartlar_enstitüsü
aop	görünüm_yönelimli_programlama
api	uygulama_programlama_arayüzü
app	uygulama_yazılımı
ascii	bilgi_değişimi_için_amerikan_standart_kodlama_sistemi
…	…

3.1.6. The Glossary of Computer Terms

In computer science, many words are used in English. So the students use these terms in their assignments even if assignments are prepared in Turkish. On the other hand, some students are careful to use Turkish meaning of these terms. This is why we created our English-Turkish computer science term list. Our computer science term list was prepared from the Informatics Association of Turkey's (TBD)

provision glossary of information terms and the computer terms provisions guide of TDK. There are 457 terms in our list. A small portion of this list is presented in Table 3.4.

Table 3.4. A Small Portion of our Computer Science Term List

English Term	**Turkish Term**
access	erişim
access_control_list	erişim_denetim_listesi
account	hesap
accuracy	doğruluk
adaptable	adapte_edilebilir
adaptable_database_system	adapte_edilebilir_işlevsel_veritabanı
address	adres
algorithm	algoritma
algorithmic_language	algoritmik_dil
…	…

3.1.7. The Turkish Stopword List

Stopwords—such as conjunctions, pronouns, prepositions, numerals, or stereotyped abbreviations— are ignored by search engines because they are progressed in almost every article and cannot provide a positive contribution to the search results. Sometimes they can negatively affect these results and cause the engine to produce wrong results. In computing, stop words are words to be sought by search engines. They are filtered out before, or after, processing of natural language text (Rajaraman and Ullman, 2011). There is not one definite stop word list to be utilized by all tools as they are selected according to a purpose.

We needed to firstly remove stopwords which are used frequently in the Turkish text or have no meaning from our data set, so we used a stopword list for Turkish. The Turkish Stopword list was obtained from the study of Can *et al.* (2008), Turkish Stop Word List of the Natural Language Processing Group of Fatih

University[3]and frequent words in our course notes. The stopword list prepared for the Turkish language is presented in Table 3.5.

Table 3.5. Turkish Stopword List

acaba	böyle	gene	kez	onlar	şeyde
altı	böylece	gibi	kırk	onlara	şeyden
altmış	bu	hâlâ	ki	onlarda	şeye
ama	bu arada	hangi	kim	onlardan	şeyi
ancak	bu yüzden	hangisi	kimde	onları	şeyler
artık	buarada	hani	kimden	onların	şimdi
asla	buna	hatta	kime	onların	şöyle
aslında	bunda	hem	kimi	onu	şu
az	bundan	henüz	kimin	onun	şuna
bana	bunu	hep	kimisi	orada	şunda
bazen	bunun	hepsi	kimse	otuz	şundan
bazı	burada	hepsinde	madem	oysa	şunlar
bazıları	burda	hepsinden	mı	oysaki	şunu
bazısı	buyüzden	hepsine	mı	öbürü	şunun
belki	bütün	hepsini	mi	ön	tabi
ben	çoğu	hepsinin	milyar	önce	tamam
bende	çoğuna	her	milyon mu	ötürü	trilyon
benden	çoğunu	her biri	mu	öyle	tüm
beni	çok	herbiri herkes	mü	rağmen	tümü
benim	çünkü	herkese	mü	sana	üç
beş	da daha	herkesi	nasıl	sanki	üzere
bile	dahi	hiç	ne	sekiz	var
bin	de	hiç kimse	ne kadar	seksen	ve
bir	defa	hiçbir	ne zaman	sen	veya
biri	değil	hiçbiri	nekadar	sana	veyahut
birçoğu	demek	hiçbirine	nezaman	sekiz	ya
birçok	diğer	hiçbirini	neden	sen	ya da
birçokları	diğeri	hiçkimse	nedir	sende	yada
biri	diğerleri diye	için	nerde	senden	yani
birisi	doksan	içinde	nerede	seni	yedi
birkaç birkaçı	dokuz	iki	nereden	senin	yerine
birkez	dolayı	ile	nereye	siz	yetmiş
birşey	dört	ise	nesi	sizde	yine
birşeyi	elbette	işte	neyse	sizden	yirmi
biz	elli	kaç	niçin	size	yoksa
bizde	en	kadar	niye	sizi	yüz
bizden	fakat	katrilyon	on	sizin	zaten
bize	falan	kendi	ona	son	zira
bizi	felan	kendine	onda	sonra	
bizim	filan	kendini	ondan	şayet	
				şey	

[3] nlp.ceng.fatih.edu.tr/blog/tr/?p=31

This stopword list was stemmed and the synonyms of these words were also added to the stopword list to improve similarity search performance in this thesis. Table 3.6 presents an example for preprocessing step in our system.

Table 3.6. A Small Portion of Preprocessed Turkish Stopword List

Original Stopword	**Stemmed Stopword**	**Synonym Stopword**
bazı	bazı	ara_sıra
bazıları	bazı	ara_sıra
bazısı	bazı	ara_sıra
…	…	…

Perl has a library for stopwords, called Lingua Stopwords. This library was maintained by Marvin Humprey and developed by Fabien Potencier. It has typical stopwords list for 12 different languages, including Danish, Dutch, English, Finnish, French, German, Hungarian, Italian, Norwegian, Portuguese, Spanish, Swedish and Russian. But it does not include Turkish. This is why we extended this library by adding the Turkish language. It has two arguments about stopwords, which are encoded in ISO code and UTF-8. So, we created our stopword list according to these arguments. For example; we transformed the word “birkaç” to the word “birkaÃ§” for UTF-8.

3.1.8. The Phrase List

In computer science, many phrases that are sequences words separated with space characters exist. So the students use these terms in their assignment, too. To improve the performance of our system, we created our phrase list. Our list was prepared both in Turkish and English. Our phrase list was prepared from the Informatics Association of Turkey’s (TBD) provision glossary of information terms, our abbreviation list, our synonyms dictionary, our glossary of computer terms, and our stopword list. There are 1052 phrases in our list. A small portion of this list is presented in Table 3.7.

Table 3.7. An Example of Our Turkish and English Computer Science Phrase List

ara sıra	arama motoru optimizasyonu
bu biçimde	bu çeşit
bunun gibi	computer architecture
çok fazla	daha çok
data base management	data base management system
data dictionary	data flow diagram
en az	en iyi
file system	file transfer protocol
first in first out	first in last out
flash memory	flaş bellek
floppy disk	flow chart
komut dili	komut dosyası
komut kümesi	komut satırı
kontrol tuşu	kullanıcı adı
open source	operating system
pek az	pek çok
pek hoş	programming language
random access memory	rastgele erişimli bellek
read only memory	salt okunur bellek
sanal bellek	tahmin etmek
tebrik etmek	ve benzerleri
veri akış diyagramı	veritabanı yöneticisi
veritabanı yönetim sistemi	veritabanı yönetimi
virtual machine	virtual memory
ya da	yönetim bilişim sistemi
…	…

3.1.9. String Matching Algorithms To Be Used

The exact string matching algorithms try to find all occurrences of the first string with in the second string. Our aim is to find all matches including exact and partial matches. We provide the detection of similarities when a change was done in the annexes of the words or the synonyms of words were used. When such changes are done, our proposed plagiarism detection system must find the matches. That's why, we cannot use the exact string matching algorithms in our thesis. We can use distance metrics or string matching algorithms which find the partial matches. According to the articles described in Previous Work section, the most successful string matching algorithms that can be used to detect textual plagiarism are Longest

Common Subsequence, Q-Gram, Winnowing, Levenshtein Distance, Cosine Similarity, Dice Similarity, Jaccard Similarity, and Greedy String Tiling Algorithms (GST). But when looking for multiple different patterns in a text, then Karp-Rabin, LCS, and GST Algorithms should be used. GST is suitable to detect the overall similarity of the two documents. GST handles the token transpositions across the two texts better than LCS. Also GST can find high similarities when the texts were reordered. Cosine, Dice and Jaccard similarities do not take into account the order of the words, unlike GST. GST is more successful than Winnowing algorithm, Q-Grams, Levenshtein Distance, Cosine Similarity, Dice Similarity, and Jaccard Similarity because the success of token based algorithms is higher than the distance based algorithms and fingerprint based algorithms according to the articles described in the Previous Work section. So, to detect the similarities between the assignments, the Cosine Similarity, and the Greedy String Tiling algorithm were used in this thesis.

Also, the other reason of using Greedy String Tiling Algorithm as the basis algorithm was that one of the most successful tools among the plagiarism detection tools, JPlag, described in the Previous Work section of this thesis use Greedy String Tiling algorithm. In our previous study (Kaya, 2011), we showed that the most successful source code plagiarism detection tools, JPlag, YAP3 and Copy/Paste Detector use Greedy String Tiling Algorithm as the basis algorithm. The source codes are actually tokens like words, so the success for the source code plagiarism detection is an important indicator for textual plagiarism detection.

In this thesis, the string matching algorithms explained in Previous Work section were tested in Results and Discussion section. According to the result of the test, the Greedy String Tiling algorithm was successful to detect similarities, although it was one of the slowest algorithms. That’s why; this algorithm was chosen to be used as the string matching algorithm to detect similarities of text lines, stopwords and punctuations in this thesis.

3.1.9.1. Cosine Similarity

Cosine Similarity is the similarity between two strings. In this thesis, the approximate string matching algorithms explained in Section 2.3.1 were tested in Section 4. According to the result of the test, the Cosine similarity was successful. That's why, the cosine similarity was chosen to be used as the approximate string matching algorithm to detect similarities in this thesis. The results of this algorithm were used as threshold for the GST based string matching algorithm.

3.1.9.2. Greedy String Tiling Algorithm

Greedy String Tiling Algorithm is a successful string matching algorithm used to compare two token strings which are essentially "Greedy String Tiling" (Wise, 1993a). It is used by JPlag Software. It gives more accurate results and it detects more successful similarities when Greedy String Tiling Algorithm is compared with the other algorithms (Schleimer *et al.*, 2003).

The purpose for which two strings, 'S1' and 'S2', are compared is to extract a set of same substrings complying with the following rules (Prechelt *et al.*, 2000; Prechelt *et al.*, 2002):

1) The rule, saying that any token of 'S1' may only be matched with exactly one token from 'S2', indicates imperfection of plagiarism programs. For it is impossible to completely match parts of the source text that have been copied into one of such programs.
2) Another rule - substrings are independent of their position in the string - suggests that reordering parts of the source text is no successful attack.
3) For the sake of proper reliability, long substring matches are preferred over short ones, as the latter ones are prone to be false.

Tokenization is an important phase of Greedy String Tiling Algorithm. After tokenization, to search for the longest contiguous matches of two files, three nested

loops are used. The first loop iterates over all the tokens in the first file, the second loop compares the token with every token in the second file. If tokens are the same, the second loop attempts to extend the match as far as possible. These two loops actions are aimed at collecting the set of all the longest common substrings. Then, all matches of maximal length are marked so that every token is only used in one match. Wise's (1993b) terminology describes a tile as a match with all the tokens marked. Minimum Match Length is defined as a lower bound for the length of a given match.

Overlapping occurs when a plagiarism tool matches the same input expressions. In order to determine which of expressions will be matched the longest possible match is performed. In such case, the first match is taken into consideration and the others are ignored. The two phases are repeatable and are conducted up to the point no further matches are available. Allowance of matches of any length would result in frequent occurrence of matches with just a few tokens.

The pseudocode of Greedy String Tiling Algorithm (Prechelt *et al.*, 2000; Prechelt *et al.*, 2002) is as shown as in Table 3.8:

Table 3.8. The Pseudocode of Greedy String Tiling Algorithm (Prechelt *et al.*, 2000; Prechelt *et al.*, 2002)

```
Greedy-String-Tiling(String A, String B) {
   tiles = {};

   do {
      maxmatch = MinimumMatchLength;
      matches = {};

      forall unmarked tokens A_a in A {
          forall unmarked tokens B_b in B {
                    j=0;
                    while ( A_{a+j} == B_{b+j} &&  unmarked(A_{a+j}) && unmarked(B_{b+j}))
                              j++;

                              if (j == maxmatch)
                                        matches = matches ⊕ match(a, b, j);
                                        if (j >maxmatch) {
                                                  matches = {match(a, b, j)};
                                                  maxmatch = j;
                                        }
                    }
          }
      }

                    forall match(a, b, maxmatch) ∈ matches {
                              for j = 0. . .(maxmatch- 1) {
                                        mark(A_{a+j});
                                        mark(B_{b+j} );
                              }
                              tiles = tiles ∪ match(a, b, maxmatch);
                    }
   } while (maxmatch > MinumumMatchLength);
   return tiles;
}
```

3.2. Method

Our plagiarism detection system designed to be indifferent to punctuation, extra whitespaces, capitalization, small matches, and permutations of a document's content. Our system handles these by pre-processing of any input text, including lowercasing, tokenization, abbreviation replacement, glossary of computer term replacement, stemming, synonym replacement and stopword removal.

The main architecture of our system is described in Figure 3.1. In Figure 3.1, "*Cos*" means the cosine score and "*Trh*" means the threshold value. Its most important design issues are related to:

- *Preprocessing:* Lowercasing, tokenization, numeric character and punctuation removal, abbreviation replacement, glossary of computer term replacement, using cosine similarity threshold, stemming, using deasciifier, synonym replacement and stopword removal;
- *Computing similarity of text lines:* using a similarity algorithm (i.e., GST) to find the longest matches of two strings;
- *Computing similarity of punctuations:* using a similarity algorithm (i.e., GST) to find the longest matches of two punctuation files;
- *Computing similarity of stopwords:* using a similarity algorithm (i.e., GST) to find the longest matches of two stopword files.

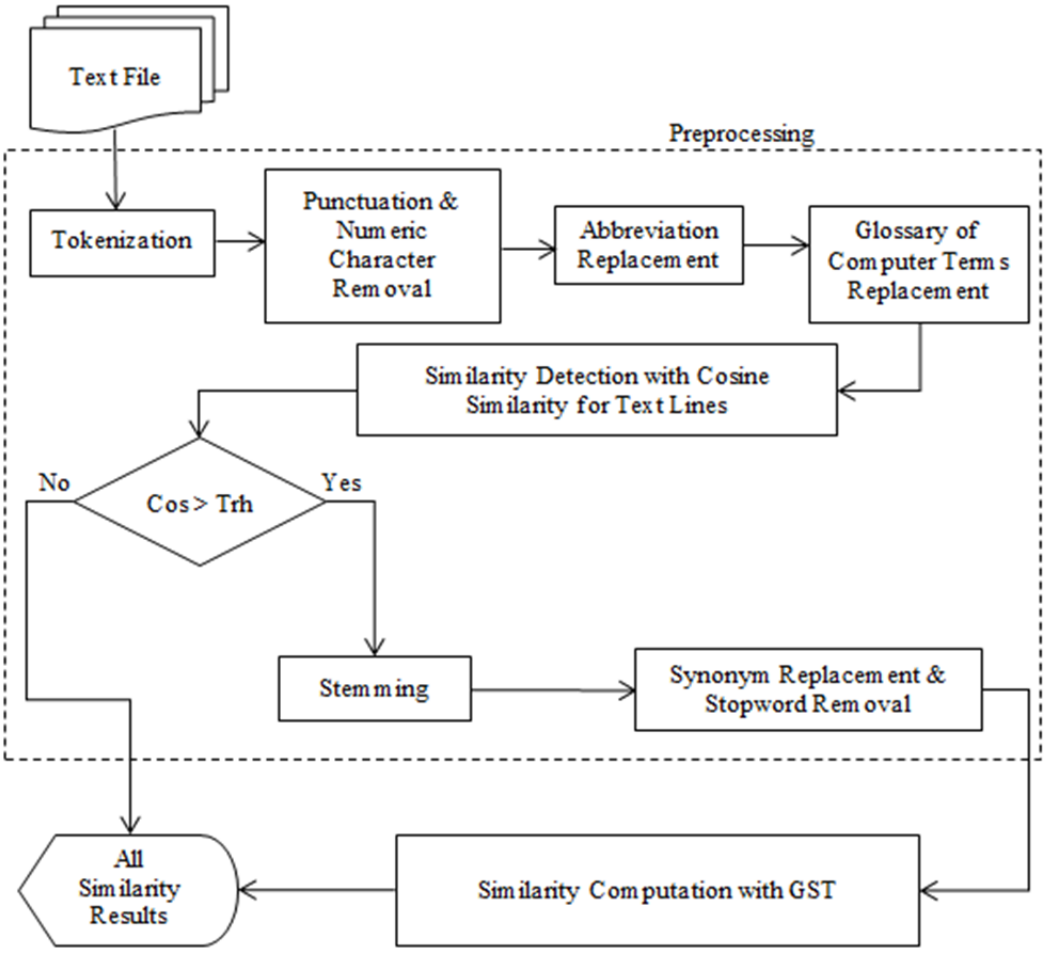


Figure 3.1. Main Architecture of the Proposed System

3.2.1. Pre-processing Phase

Most methods include a pre-processing phase in which stopword and punctuations are removed and words are stemmed. Transformation of a Turkish text into a structured form, so it could be convenient for plagiarism detection, requires undertaking of several steps.

- *Converting the Format:* The document is converted to plain text.
- *Lowercasing:* All letters in the text are converted to lowercase.
- *Punctuation Removal:* All punctuations and special characters in the text are removed, and the punctuations are stored in another file to compare similarities of files by using punctuations.
- *Extra Whitespaces and Numeric Character Removal:* Multiple spaces are discarded and the words are extracted. Extra whitespace characters are converted into a single space and numbers are removed from the text.
- *Abbreviation Replacement:* The abbreviations of the words in the text are converted into their equivalent of written-out words.
- *Using Glossary of Computer Terms:* The words of English in the text are converted into their Turkish meanings.
- *Tokenization:* Input text is split up into tokens.
- *Cosine Similarity Threshold:* The cosine similarity threshold is used for our similarity method. The result of Cosine Similarity is used for preselection as a threshold value. It is not used as a string matching algorithm alone, because if a long document is compared with a short document, the cosine similarity is lower than expected. If cosine similarity of documents is lower than the threshold, similarity computations with GST are not performed to save time, since it is highly possible that these documents are not similar.
- *Stemming:* Zemberek is used to reduce words to their root by removing suffixes and prefixes. Also, the module of Zemberek, a Turkish text deasciifier, is used to retain accented characters from non-accented Turkish

text. This feature converts a Turkish text without Turkish letters into a text with proper Turkish letters.

- *Synonym Replacement:* The words in the text are converted into their most frequent synonyms. This feature may help to detect advanced forms of hidden plagiarism.
- *Stopword Removal:* The stopwords are removed from the text in order to get more significant results. These stopwords are inserted into another file, to compare files according to stopwords by using GST algorithm.

Table 3.9 presents an example for preprocessing steps of a sentence in our system.

Table 3.9. The Preprocessing Steps for a Sentence in Our System

Turkish Text	→	1- RAM, bir Hardware parcasidir.
Lowercasing	→	1- ram, bir hardware parcasidir.
Punctuation Removal	→	1 ram bir hardware parcasidir
Extra Whitespaces and Numeric Character Removal	→	ram bir hardware parcasidir
Abbreviation Replacement	→	rastgele erişilebilir bellek bir hardware parcasidir
Using Glossary of Computer Terms	→	rastgele erişilebilir bellek bir donanım parcasidir
Tokenization	→	rastgele erişilebilir bellek bir donanım parcasidir
Using Deasciifier & Stemming	→	rastgele erişilebilir bellek bir donanım parçasıdır rastgele eriş bellek bir donanım parça
Synonym Replacement	→	gelişigüzel eriş akıl bir donanım bölüm
Stopword Removal	→	gelişigüzel eriş akıl donanım bölüm

3.2.2. Similarity Phase

Many similarity metrics, including the string matching algorithms described in the Preliminary Work section, exist for similarity detection.

In this study, we planned to use an algorithm based on Greedy String Tiling for plagiarism detection in Turkish documents. GST was selected as the basis algorithm in this study because it gives more accurate results with the test data and it

detects more successful similarities. Also, Greedy String Tiling algorithm has not been used for the Turkish language among the string matching algorithms. In terms of the originality and the success of the proposed thesis, it was indicated that the Greedy String Tiling algorithm for the similarity detection of documents would be useful. In our previous study, Greedy String Tiling algorithm provided high performance for the detection of source code similarity in programming languages assignments (Kaya, 2011).

3.2.2.1. GST Based Text Line Matching Algorithm

Greedy String Tiling Algorithm was used for making comparisons in the Document Similarity Algorithm. It works on only the words without whitespace characters, numbers, punctuation and stopwords, which are removed from the file. The remaining words are converted into tokens. For every unmarked token in the first file, the algorithm begins by looking for an identical token in the second file. In this algorithm, minimum match length is taken as a value of 1 and 2. The tokens comparison serves towards expanding the match as far as possible. When one of the tokens of the two files is marked, a maximum match is found. Then a similarity rate is produced and is written to a database to be weighted when all algorithms are finished. The processing time is calculated and written to the database to compute total processing time of our system.

3.2.2.2. GST Based Punctuation Matching Algorithm

Greedy String Tiling Algorithm is used for making comparison in the Punctuation Matching Algorithm. It works on only punctuation characters of the file. The whitespace characters, numbers and the words are removed from the file. The same processes in GST Based Text Line Matching Algorithm are used in this algorithm, too. In this algorithm, minimum match length is taken as a value of 1. The tokens comparison serves towards expanding the match as far as possible. When one of the tokens of the two files is marked, a maximum match is found. Then a

similarity rate is produced and it is written to the database to be weighted when all algorithms are finished. The processing time is calculated and written to the database as well.

3.2.2.3. GST Based Stopword Matching Algorithm

Greedy String Tiling Algorithm is used for the comparison in the Stopword Matching Algorithm. It consists of only Turkish stopwords of the documents. The punctuation and whitespace characters, all tokens except stopwords are removed from the file. Then, the GST algorithm is applied to stopwords sequences of the two files. In this algorithm, minimum match length is taken as 1. This algorithm tries to find the first matching stopwords, and then the words are served towards expanding the match as far as possible. When one of the words of the two files is marked, a maximum match is found. It produces a score and it is written to the database to be weighted when all algorithms are finished. The processing time is calculated and written to the database as well.

3.2.2.4. Computation of Similarity Score

The GST Based Text Line Matching Algorithm, GST Based Punctuation Matching Algorithm and GST Based Stopword Matching Algorithm produce a score. A single score is assigned for the similarity of the document pairs. If a document pair has a higher score, it is implied that these documents are more similar and may be plagiarized from each other or from a common third document. The score of each individual algorithm is weighted and the Total Match Score is calculated according to Equation 3.1. These weights are adjusted to give optimal results. α is the weight coefficient, GBSMA is the score of the Greedy String Tiling Based Stopword Matching Algorithm, GBTMA is the score of the Greedy String Tiling Based Text Line Matching Algorithm, GBPMA represents the score computed by the Greedy String Tiling Based Punctuation Matching Algorithm on Equation 3.1. The scores of algorithms are calculated according to Equation 3.3 and 3.4.

$$\text{TotalMatchScore} = \alpha_1.\text{GBTMA} + \alpha_2.\text{GBSMA} + \alpha_3.\text{GBPMA} \quad (3.1)$$

$$\alpha_1 + \alpha_2 + \alpha_3 = 1 \quad (3.2)$$

$$similarity_score(A, B) = \frac{2 * coverage(tiles)}{|A| + |B|} \quad (3.3)$$

$$covarage(tiles) = \sum_{match(a,b,length) \in tiles} length \quad (3.4)$$

These similarity score are average scores of two files. Also, the individual scores are calculated according to Equation 3.4, 3.5 and 3.6.

$$similarity_score(A\ to\ B) = \frac{coverage(tiles)}{|A|} \quad (3.5)$$

$$similarity_score(B\ to\ A) = \frac{coverage(tiles)}{|B|} \quad (3.6)$$

If the similarity score generated by the GST Based Text Line Matching Algorithm is greater than a threshold value, then we consider similarity scores produced by GST Based Stopword Matching and GST Based Punctuation Matching algorithms to determine the severity of the plagiarism. Otherwise, we use 1 for α_1, and other α coefficients are set to 0, because it is possible that non-similar documents may contain similar punctuations or similar stopwords.

In the end of this study, a HTML output report with a list of file pairs ordered by their match scores is produced. Figure 3.2 shows the flowchart of the proposed similarity computation system.

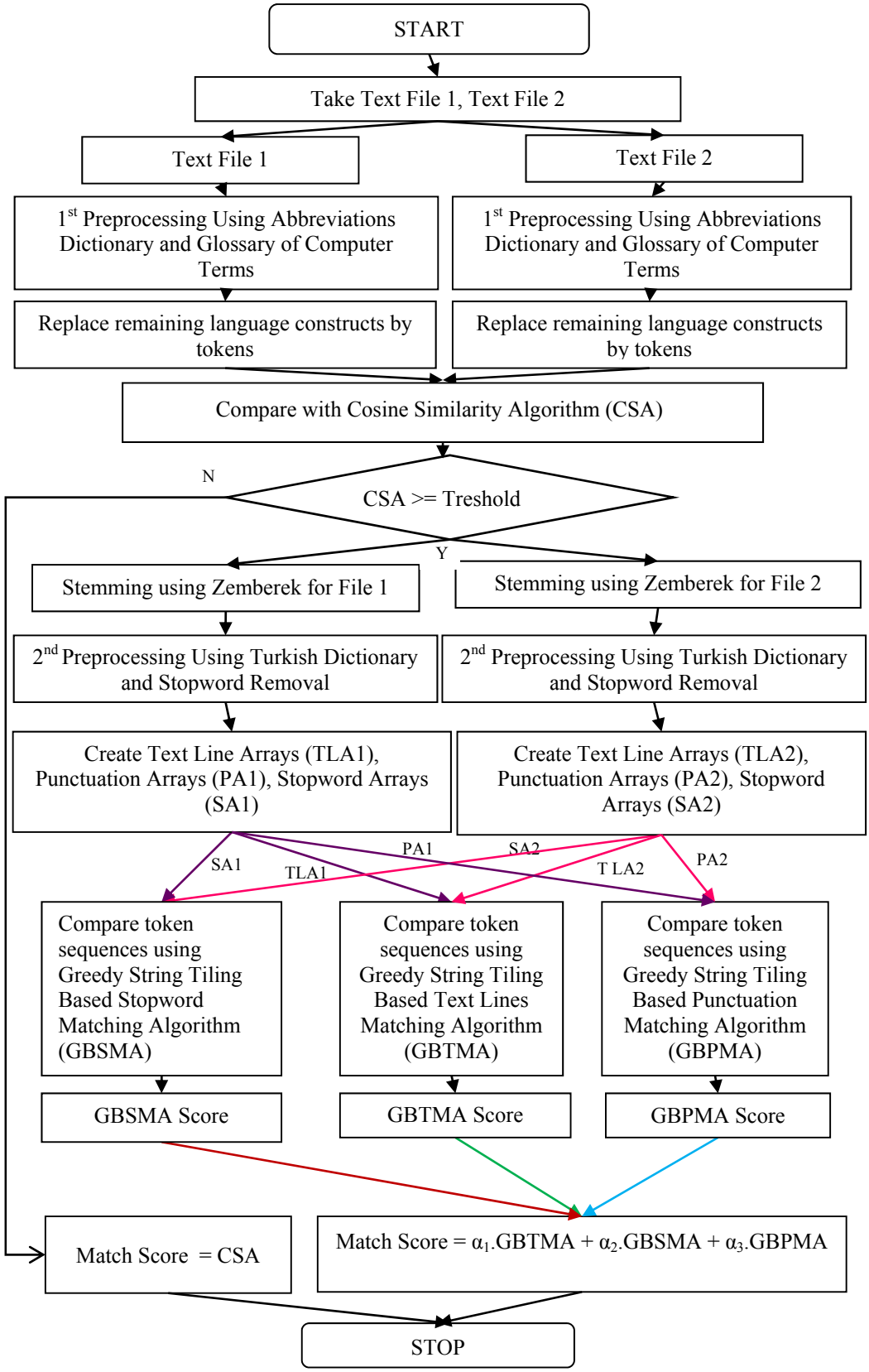


Figure 3.2. The Proposed String Matching Algorithm

3.2.2.5. Integration of the Proposed System into Moodle Environment

Our plagiarism detection system was built in Moodle. To integrate the plagiarism detection tool to the Moodle system, we updated the source codes of Moodle, and we included a new button "Metin Karsilastir" (Compare Texts) to the user interface. When students submit their assignments to Moodle as shown in Figure 3.3, teachers can read them one by one or in the lump. In order to control all assignments in the lump, first "… ödev gönderisine bak" link should be clicked in Figure 3.3, and then the "Metin Karsilastir" button as shown in Figure 3.4 is used for comparing the submitted assignments with our algorithm, which consists of Greedy String Tiling Based Text Line Matching, Greedy String Tiling Based Punctuation Matching and Greedy String Tiling Based Stopword Matching Algorithms.

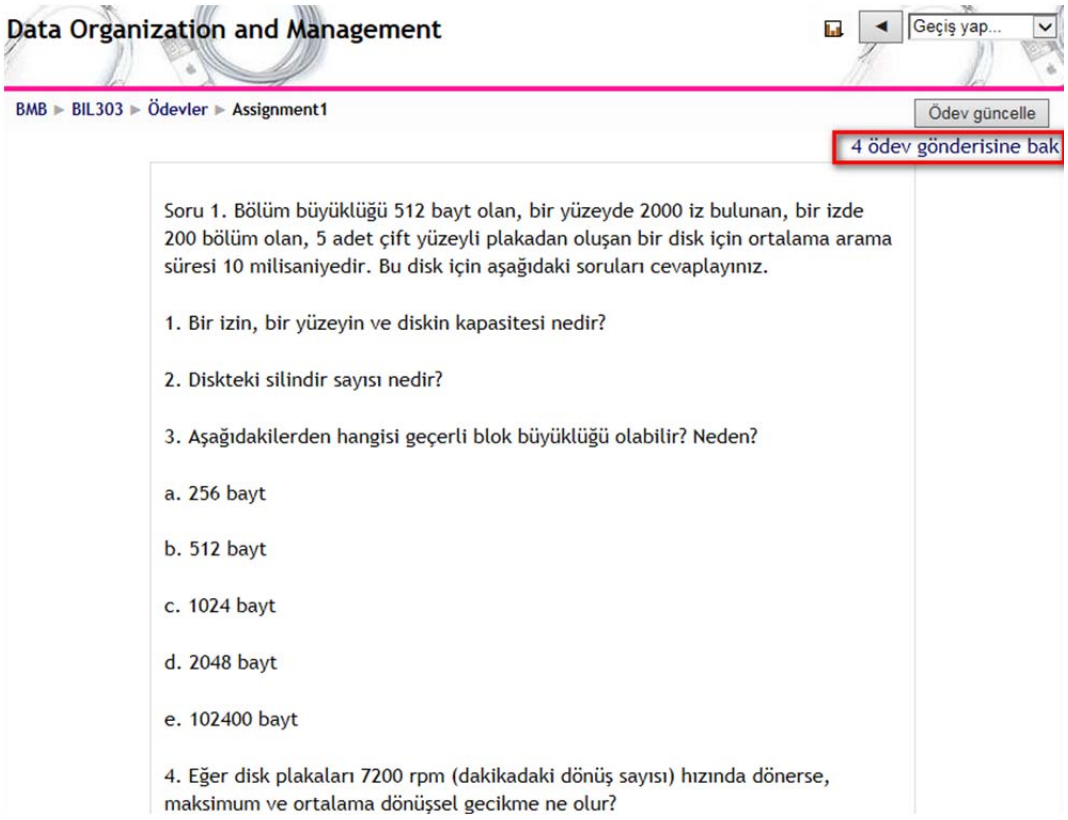


Figure 3.3. Moodle Web Page of Submitted Assignment

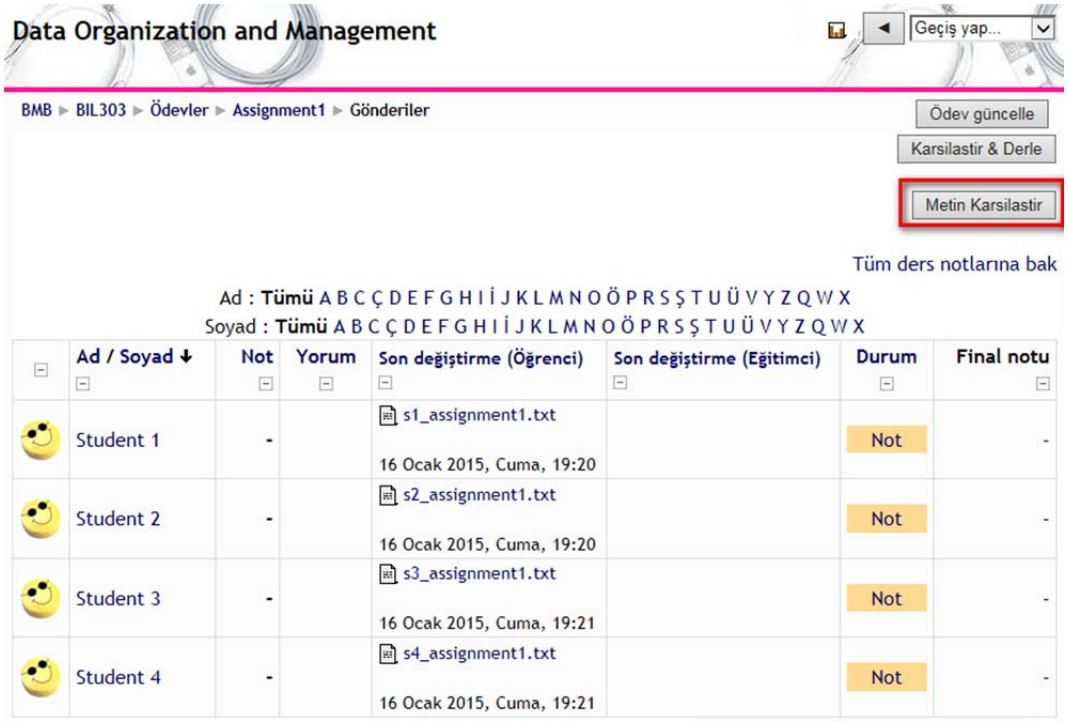
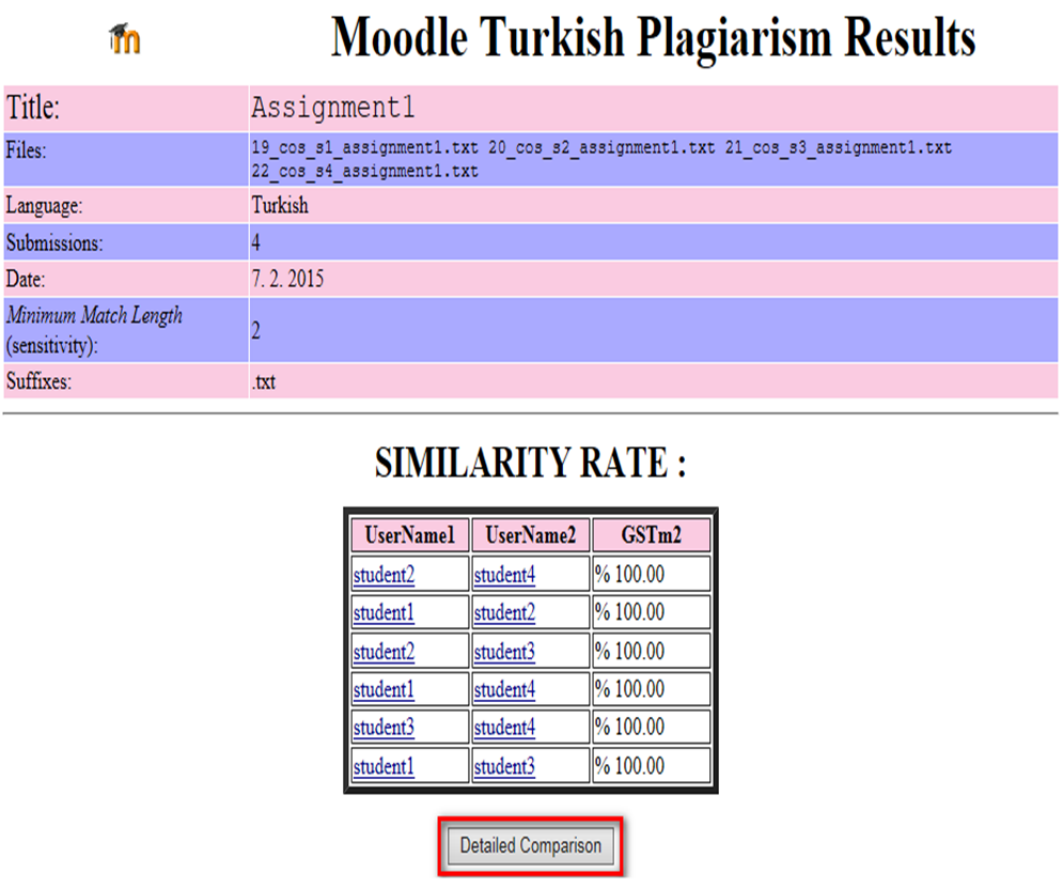


Figure 3.4. Moodle Detailed Web Page of Submitted Assignment

As shown in Figure 3.5, on the top of the page, there is a table which shows the name of the assignment, the number of submissions, the name of submitted assignments, the date, and the minimum match length (the sensitivity of greedy string tiling algorithm).

For example, if *n* assignments are loaded to the Moodle system, our plagiarism detection tool compares each pair of submitted text files, computes similarity scores, and stores the similarity values within the database.



Moodle Turkish Plagiarism Results

Title:	Assignment1
Files:	19_cos_s1_assignment1.txt 20_cos_s2_assignment1.txt 21_cos_s3_assignment1.txt 22_cos_s4_assignment1.txt
Language:	Turkish
Submissions:	4
Date:	7. 2. 2015
Minimum Match Length (sensitivity):	2
Suffixes:	.txt

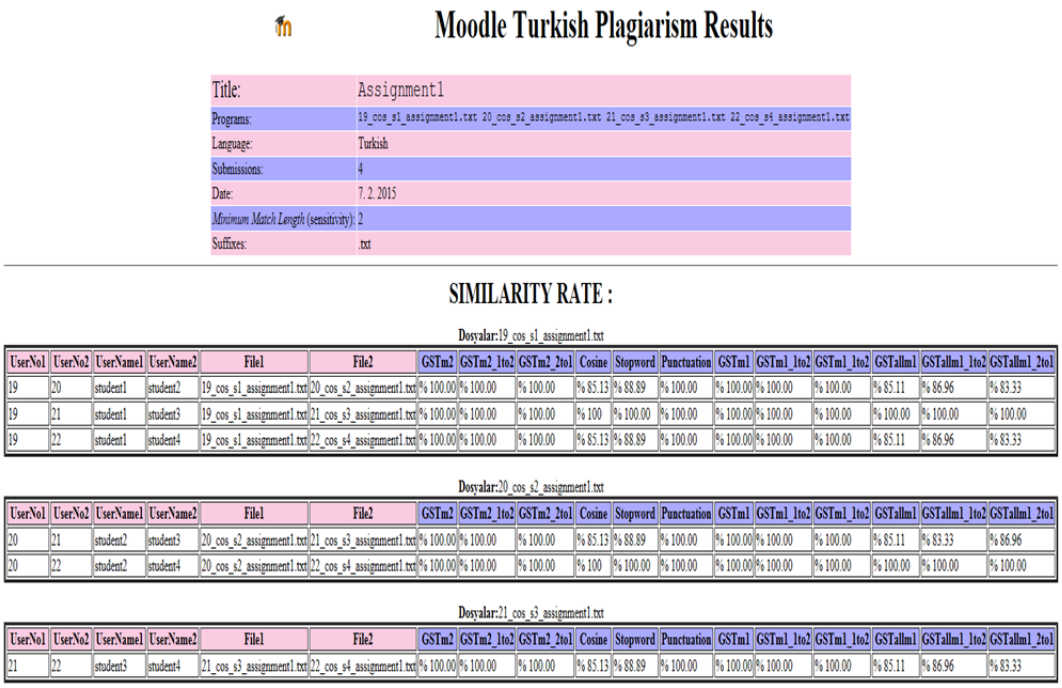
SIMILARITY RATE :

UserName1	UserName2	GSTm2
student2	student4	% 100.00
student1	student2	% 100.00
student2	student3	% 100.00
student1	student4	% 100.00
student3	student4	% 100.00
student1	student3	% 100.00

Detailed Comparison

Figure 3.5. The Result Web Page of the Proposed Algorithm

To view the detailed result of similarities as shown in Figure 3.6, user can click “Detailed Comparison” button as shown in Figure 3.5. It shows that the assignments are grouped and compared with each other. The similarity rate of text lines, the similarity rate of stopwords, and the similarity rate of punctuations are showed in Figure 3.6.



Moodle Turkish Plagiarism Results

Title:	Assignment1
Programs:	19_cos_s1_assignment1.txt 20_cos_s2_assignment1.txt 21_cos_s3_assignment1.txt 22_cos_s4_assignment1.txt
Language:	Turkish
Submissions:	4
Date:	7. 2. 2015
Minimum Match Length (sensitivity):	2
Suffixes:	.txt

SIMILARITY RATE :

Dosyalar:19_cos_s1_assignment1.txt

UserNo1	UserNo2	UserName1	UserName2	File1	File2	GSTm2	GSTm2_1to2	GSTm2_2to1	Cosine	Stopword	Punctuation	GSTm1	GSTm1_1to2	GSTm1_1to2	GSTallm1	GSTallm1_1to2	GSTallm1_2to1
19	20	student1	student2	19_cos_s1_assignment1.txt	20_cos_s2_assignment1.txt	% 100.00	% 100.00	% 100.00	% 85.13	% 88.89	% 100.00	% 100.00	% 100.00	% 100.00	% 85.11	% 86.96	% 83.33
19	21	student1	student3	19_cos_s1_assignment1.txt	21_cos_s3_assignment1.txt	% 100.00	% 100.00	% 100.00	% 100	% 100.00	% 100.00	% 100.00	% 100.00	% 100.00	% 100.00	% 100.00	% 100.00
19	22	student1	student4	19_cos_s1_assignment1.txt	22_cos_s4_assignment1.txt	% 100.00	% 100.00	% 100.00	% 85.13	% 88.89	% 100.00	% 100.00	% 100.00	% 100.00	% 85.11	% 86.96	% 83.33

Dosyalar:20_cos_s2_assignment1.txt

UserNo1	UserNo2	UserName1	UserName2	File1	File2	GSTm2	GSTm2_1to2	GSTm2_2to1	Cosine	Stopword	Punctuation	GSTm1	GSTm1_1to2	GSTm1_1to2	GSTallm1	GSTallm1_1to2	GSTallm1_2to1
20	21	student2	student3	20_cos_s2_assignment1.txt	21_cos_s3_assignment1.txt	% 100.00	% 100.00	% 100.00	% 85.13	% 88.89	% 100.00	% 100.00	% 100.00	% 100.00	% 85.11	% 83.33	% 86.96
20	22	student2	student4	20_cos_s2_assignment1.txt	22_cos_s4_assignment1.txt	% 100.00	% 100.00	% 100.00	% 100	% 100.00	% 100.00	% 100.00	% 100.00	% 100.00	% 100.00	% 100.00	% 100.00

Dosyalar:21_cos_s3_assignment1.txt

UserNo1	UserNo2	UserName1	UserName2	File1	File2	GSTm2	GSTm2_1to2	GSTm2_2to1	Cosine	Stopword	Punctuation	GSTm1	GSTm1_1to2	GSTm1_1to2	GSTallm1	GSTallm1_1to2	GSTallm1_2to1
21	22	student3	student4	21_cos_s3_assignment1.txt	22_cos_s4_assignment1.txt	% 100.00	% 100.00	% 100.00	% 85.13	% 88.89	% 100.00	% 100.00	% 100.00	% 100.00	% 85.11	% 86.96	% 83.33

Figure 3.6. The Detailed Similarity Report

The numbers of matched tokens for every file are calculated as shown as in Figure 3.7 and written to the database.

NUMBERS of MATCHED TOKENS:

UserNo1	UserName1	UserNo2	UserName2	GSTToken	CosineToken	StopToken	PuncToken	GST2Token	GSTm1Token
21	student3	22	student4	21	20	4	1	21	20
20	student2	21	student3	21	20	4	1	21	20
20	student2	22	student4	21	24	5	1	21	24
19	student1	20	student2	21	20	4	1	21	20
19	student1	21	student3	21	23	4	1	21	23
19	student1	22	student4	21	20	4	1	21	20

Figure 3.7. Numbers of Matched Tokens

The process time of the proposed algorithm is shown as in Figure 3.8. Each time of the preprocessing steps are located in separate columns. GSTTime is the process time of the document comparison when the minimum match length was taken as 2. Token1Time is the process time of the first preprocessing step, when Token2Time is the process time of the last preprocessing step. ZemberekTime is the process time of the stemming. StopTime is the process time of the stopwords

comparison, when PuncTime is the process time of the punctuation comparison. GST2Time is the process time of the document comparison when the minimum match length was taken as 1. GSTm1Time is the process time of the document comparison when the minimum match length was taken as 1 and no preprocessing was done.

PROCESS TIME:

GSTTime	CosineTime	StopTime	PuncTime	Token1Time	Token2Time	ZemberekTime	GST2Time	GSTm1Time	TimeScore
22.922 ms	23.002 ms	21.437 ms	19.361 ms	340.504 ms	480.52 ms	468 ms	19.466 ms	20.869 ms	1416.081 ms

Figure 3.8. Processing Time of the Algorithms

3.2.3. Web Usage Mining on Moodle Logs

In this study, our secondary aim is to improve the plagiarism detection by Web usage mining. Web Usage Mining aims to produce knowledge from the data left behind by site visitors. For this purpose, it uses data mining techniques to extract meaningful data and discover the most intense and interesting user access patterns from log files (Özseven and Düğenci, 2011; Kantardzic, 2003). Web Usage Mining is performed in 3 stages, including pre-processing, pattern discovery and pattern analysis (Özseven and Düğenci, 2011; Kosala and Blockeel, 2000).

In this study, the connection between student behavior in the logs of the Moodle e-learning system and the cheating behavior is analyzed by some web usage mining methods. For this purpose, we use Weka data mining tool, and classification and association rule analysis methods from this tool. Association Rule Mining discovers associations between attribute values of data set (Thabtah *et al.*, 2006). This technique is one of the most important and well-researched techniques of data mining (Kotsiantis and Kanellopoulos, 2006). The Apriori Algorithm (Han and Kamber, 2006), the most preferred and widely-used association algorithm, was developed by Agrawal and Srikant (1994). Each rule obtained from the results of Apriori Algorithm was expressed with support criteria— the frequency of association

between items— and confidence criteria—the accuracy of this association (Keleş and Kaya, 2014). It was also used to highlight the general trend in the dataset (Kaya *et al.*, 2013). Class association rules (CAR) was planned to be used instead of the general association rules in this study. Mining class association rules (CAR) find all possible rules that exist within data. This special subset of rules is used for classification (Liu *et al.*, 1998).

The data for the analysis was collected from the Data Organization and Management Moodle Course at Çukurova University Computer Engineering Department during the 2013-2014 semesters. The mdl_assignment, mdl_assignment_submission, mdl_course, mdl_course_modules, mdl_grade_grades, mdl_log, mdl_log_display, and mdl_user tables from Moodle were used. Thirty one attributes were used, including identification number of student, identification number of assignment, new student info, number of login, number of assignment uploaded, assignment upload date before deadline, assignment view date after assignment given date, number of assignment viewed, number of total resource viewed, course enroll date, assignment grade, midterm scores, final scores, class grade, number of sentence used, number of word used, number of average word used, number of different word used, number of period used, number of comma used, number of line, number of semicolon used, number of question mark used, number of exclamation used, number of noun used, number of verb used, number of adjective used, number of pronoun used, number of preposition used, number of number of conjunction used, and plagiarism info.

Collected logs in this thesis were preprocessed using the Knowledge Extraction based on Evolutionary Learning (Keel) program and classified using the Waikato Environment for Knowledge Analysis (Weka) program[4]. Weka is based on Java with open source code and includes data preprocessing, classification, clustering, association rule mining, attribute selection and visual interactive page. Keel is based on Java with open source code and includes regression, classification, clustering, pattern mining, training set selection, feature selection, discretization, and imputation methods for missing values.

[4] http://www.cs.waikato.ac.nz/ml/weka/

In the first stage, student logs were pre-processed and made consistent and meaningful. For instance, they were normalized with Min-Max Normalization and discretized with Uniform Frequency using Keel. The Min-Max Normalization method transforms a value *v* to *v'* which fits in the range [*new_min, new_max*]. We take *new_min* as 0 and *new_max* as 1. The formula of the Min-Max Normalization method is given in Equation 3.7.

$$v' = \frac{v - min_A}{max_A - min_A}(new_max_A - new_min_A) + new_min_A \quad (3.7)$$

The Uniform Frequency discretization method divides the range of the attributes into *N* intervals. Each interval contains approximately same number of samples. The reason of using the Uniform Frequency method is that this method has the ability of skewed data handling.

After the pre-processing stage, classification and association rule analysis were applied to improve plagiarism detection process, and to analyze behaviors of students who tend to plagiarism. Thus, we examined whether the user behavior affects the plagiarism detection or not, and we observed common properties of cheating students. For instance, when a student never logged into the Moodle e-learning system, never studied lecture notes before loading the assignment, then it can be concluded that this student probably tends to cheating. In the last stage, F-measure, a harmonic mean of precision and recall, were calculated to measure the performance of the web usage mining on test data rules. Precision, recall, accuracy and F-Measure are then defined as shown in Table 3.10 and Equation 3.8, 3.9, 3.10 and 3.11 (Han and Kamber, 2006; Liu, 2011; Mundluru, 2008).

Table 3.10. The 2x2 Confusion Matrix In Classification

	Classified Positive	**Classified Negative**
Actual Positive	TP	FN
Actual Negative	FP	TN

$$Precision = \frac{TP}{TP + FP} \tag{3.8}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.9}$$

$$Accuracy = \frac{Number\ of\ Correct\ Classifications}{Total\ Number\ of\ Test\ Cases} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.10}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.11}$$

A Confusion Matrix for one more than class in the dataset (Witten and Frank, 2005) is shown in Table 3.11.

Table 3.11. A Confusion Matrix for One More than Class

	PREDICTED CLASS			
KNOWN CLASS LABEL IN DATA		**A**	**B**	**C**
	A	TP_A	E_{AB}	E_{AC}
	B	E_{BA}	TP_B	E_{BC}
	C	E_{CA}	E_{CB}	TP_C

For the precision and recall of the class *A* in Table 3.11 is used and the values are calculated as following:

$$Precision_A = \frac{TP_A}{TP_A + E_{BA} + E_{CA}} \tag{3.12}$$

$$Recall_A = \frac{TP_A}{TP_A + E_{AB} + E_{AC}} \tag{3.13}$$

4. RESULTS AND DISCUSSION

In this section experiments performed and their results are presented. Perl programming language[1] was used for implementing the algorithms.

The proposed method was tested under Microsoft Windows 7 Ultimate 64-bit operating system. The hardware used in the experiments had 16 GB of RAM and Intel® Xeon® E5-2600 v2 processor. The proposed method was tested on the Data Organization and Management Course assignments in Computer Engineering Department of Çukurova University, some datasets from YTÜ Kemik Group, a book of Kafka, and some movie web sites. Aside from these datasets, other small datasets were used for determining the appropriate string matching algorithms which were the base of our hybrid algorithm. In this study to run Apache[2], MySql[3], Perl source codes and PHP[4] source codes, XAMPP[5] program was installed on a computer. To run Perl on a Windows computer, Strawberry Perl[6] was installed.

Detailed information about the algorithms was given in the previous section. The proposed method has been compared with for SIM, Sherlock, WCopyfind, CopyCatchGold, Ferret, DOCCop, Turnitin and Text_Matching Software of YTÜ, Document Similarity Software of YTÜ, and our implemented Cosine, Dice, and Jaccard text similarity measures in a number of experiments.

4.1. Experiments for Determining the Need of Plagiarism Detection Tool Usage

Before developing our plagiarism detection tool, we performed an experiment to determine whether such a tool is necessary or not. So, in this experiment, our aim is to show the effect of using a plagiarism detection tool on giving similar assignments behavior of our students. For this experiment, firstly we used the submissions made by our students for the assignments of the Data Structures course

[1] http://www.perl.org/
[2] http://www.apache.org/
[3] http://www.mysql.com/
[4] http://www.php.net/
[5] http://www.apachefriends.org/en/xampp.html
[6] http://strawberryperl.com/

which was offered in 2010 Fall, and 2011 Fall semesters (Kaya and Özel, 2014a) and we included Moss plagiarism detection tool into our Moodle system. These results are not an evidence for plagiarism, these are only similarity rates. So, human judgment is required to determine if an instance of plagiarism has occurred when the similarity rates are high.

The Data Structures course was offered by the same lecturer in both 2010 and 2011 Fall semesters. All the assignments given in this course were prepared and graded by the same lecturer. The given assignments were different from each other in each semester and they have similar difficulty levels. In the 2010 Fall semester, our Moodle system was not enriched with the Moss system, and we did not inform our students about the usage of source code plagiarism detection tool. We gave three programming assignments in the 2010 Fall semester. We have checked these submissions for plagiarism 12 months later after we have included the Moss plagiarism detection system into our Moodle web site (Kaya and Özel, 2014a). The results are presented in Table 4.1.

Table 4.1. Students' Plagiarism Behavior for Assignments in 2010 Fall Semester

Similarity Rate	**Assignment 1**	**Assignment 2**	**Assignment 3**
0-50%	76 %	73.6 %	63.6 %
51-70%	21.6 %	4 %	8.8 %
71-100%	2.4 %	22.4 %	27.6 %

The similarity rate up to 50% is considered as "no plagiarism" by us, because our students need to use common function definitions (i.e., source codes) to do the assignments. Similarity rate between 51–70% and 71–100% are named as "moderate plagiarism", and "complete plagiarism" respectively. Also, each assignment is more difficult than the previous assignments. According to Table 4.1, as the assignments get harder, our students tend to send similar assignments since students do not have any information about the presence of successful plagiarism detection tools. We also performed a t-test to compare plagiarism behavior of students against different assignments. For the 2010 Fall semester, there is no statistically significant difference between plagiarism behavior of students for assignments 1 and 2 ($p = 1.000 > 0.05$); however there is a statistically significant difference between

assignments 1 and 3 ($p = 0.034 < 0.05$) since assignment 3 is harder than assignment 1.

At the beginning of the 2011 Fall semester, we informed our students about the Moss plagiarism detection tool. We said to our students that we would use this tool within our Moodle system and we again gave three assignments having similar difficulties with the previous year's assignments. For each assignment, we gave feedbacks to students about their submitting similar assignments behavior through our Moodle system. The assignment grades of the students whose assignments had a similarity rate were decreased (Kaya and Özel, 2014a). The submitting similar assignments behavior of our students for 2011 Fall semester is given in Table 4.2.

Table 4.2. Students' Plagiarism Behavior for Assignments in 2011 Fall Semester

Similarity Rate	**Assignment 1**	**Assignment 2**	**Assignment 3**
0-50%	96.4 %	92.8 %	93.08 %
51-70%	2 %	2.4 %	1.89 %
71-100%	1.6 %	4.8 %	5.03 %

As it can be easily seen from Table 4.2, giving information about the usage of the plagiarism detection tool to our students prevents them from submitting similar assignments. Even for hard assignments (2 and 3) more than 92% of our students submitted their own works; and this result made us happy since we, as instructors, want to see our students' own performance when we gave assignments to them. For the 2011 Fall semester, there is no statistically significant difference between plagiarism behavior of students for assignments 1 and 2 ($p = 0.300 > 0.05$); for assignments 1 and 3 ($p = 0.658 > 0.05$); and for assignments 2 and 3 ($p = 0.858 > 0.05$). This result shows that giving information about the usage of the plagiarism detection tool to our students prevents them from submitting similar assignments even for hard assignments.

We also compared plagiarism behavior of students for the 2010 and 2011 Fall semesters by using independent samples t-test. Between the first assignments of 2010 and 2011 Fall semesters there is no statistically significant difference in terms of the plagiarism behavior ($p = 0.166 > 0.05$). This result sound meaningful since the first assignments were easy and most students can do them easily by themselves.

However, it was observed that there is a statistically significant difference between the 2010 and 2011 Fall semesters for the second (p = 0.001) and the third (p=0.000) assignments in terms of the plagiarism behavior which again shows that even for hard assignments, using plagiarism detection tool prevents our students from submitting similar assignments.

In this experiment we also show the effect of using a plagiarism detection tool on the success of our students. In Table 4.3 percentage of letter grades given to our students for the Data Structures course offered in 2010 Fall, and 2011 Fall semesters are listed.

Table 4.3. Students' Performances for the Data Structures Course in 2010 and 2011 Fall Semesters

Letter Grade	**Weight of Letter Grade (Out of 4.00)**	**% of Students for 2010 Fall Semester**	**% of Students for 2011 Fall Semester**
AA	4.00	0.6	5.6
BA	3.50	5.6	6.3
BB	3.00	3.1	9.1
CB	2.50	6.8	6.3
CC	2.00	23.0	14.0
DC	1.50	14.3	16.1
DD	1.00	16.8	21.0
FD	0.50	14.9	6.3
FF	0	10.6	11.2
NA	-	4.3	4.1
TOTAL		100	100

Letter grades AA to CC that includes grades AA, BA, BB, CB, CC have weights 4, 3.5, 3.0, 2.5, and 2.0 out of 4.0 respectively, are given to the successful students whose weighted averages are above or equal to the average of the class. Letter grades DC and DD are given to students whose weighted averages are near but below to the average of the class; and these grades means conditional pass from the class. Letter grades FD and FF mean fail from the class. NA grade also means fail, but it is given to students who do not attend the class.

As shown in Table 4.3, the percentage of our successful students increases in 2011 Fall semester. For example in 2010 Fall semester less than 1% of the students got AA from the class, while in 2011 Fall semester 5.6% of the students got AA from the same class. Also in 2010 Fall semester, 29.8% of the students failed from the

class by getting FD, FF, and NA grades. However in 2011 Fall semester, 21.6% of the students failed from the class, so the percentage of failing decreased by 8%. We also made independent samples t-test to show the statistical difference between the grades for the 2010 and 2011 Fall semesters. According to the t-test, there is a statistically significant difference ($p = 0.037$) between the grades. It is also observed that the mean of the grades was 1.4 out of 4.00 for 2010 Fall semester, and the mean value increased to 1.64 out of 4.00 in 2011 Fall semester, which shows that using the plagiarism detection embedded Moodle system enforces our students to work harder to do their assignments and effects positively their success rate. Since Moss plagiarism detection tool is used for source codes, to check similarities for natural language texts, we need efficient plagiarism detection tools and include them into the Moodle system.

As an example, Data Organization and Management course, which is offered in our department, includes assignments to be written in Turkish. Since Moss is used to detect plagiarism in source codes, we could not use it to detect similarities in student assignments for this course. When we checked the student assignments submitted for this course in 2012 and 2013 Fall semesters, after developing our plagiarism detection tool, the plagiarism behavior of our students are presented in Table 4.4 and 4.5.

In the 2012 Fall semester, our Moodle system was not enriched with our proposed plagiarism detection system. At the beginning of the 2012 Fall semester, we said to our students that we could check for plagiarism manually and after the second assignment we gave manual feedbacks to them about their plagiarism behavior through our Moodle system. We decreased the grades of some students who had a high similarity rate by detected manually. The observed plagiarism results are presented in Table 4.4.

Table 4.4. Students' Plagiarism Behavior for Assignments in 2012 Fall Semester

Similarity Rate	**A1**	**A2**	**A3**	**A4**
0-50%	0 %	0 %	6.25 %	10 %
51-70%	5.41 %	2.63 %	3.13 %	6.67 %
71-100%	94.59 %	97.37 %	90.63 %	83.33 %

A: Assignment

According to Table 4.4, the assignments' similarity rates of more than 90% of our students were high because we could only detect very small portion of similarities. For the 2012 Fall semester, there is no statistically significant difference between plagiarism behavior of students for assignments 1 and 2 ($p = 0.450 > 0.05$); for assignments 1 and 3 ($p=0.516$); for assignments 1 and 4 ($p = 0.784 > 0.05$); and there is no statistically significant difference between assignments 2 and 3 ($p=0.931$); for assignments 2 and 4 ($p=0.626$); and there is no statistically significant difference between assignments 3 and 4 ($p=0.659$). As shown in Table 4.4, after giving feedback to students, plagiarism rate decreased slowly since only small portion of plagiarized students could be detected manually.

At the beginning of the 2013 Fall semester same as the 2012 Fall semester, we said to our students that we could check plagiarism manually and decrease the grades of students who had a high similarity rate. We gave five assignments having similar difficulties with the previous year's assignments. After the second assignment in 2013 Fall semester, we gave feedbacks to them about their plagiarism behavior through our Moodle system. We informed our students late because it takes too much time to evaluate assignments with manual methods. We decreased the grades of students whose assignments had a high similarity rate that were detected manually. The plagiarism behavior of our students for 2013 Fall semester is given in Table 4.5. When we performed t-test for the five assignments, we observed that there is no statistically significiant difference between the students' behavior for the assignments. According to Table 4.5, the complete plagiarism decreases from 91.30% to 54.55%, especially, after the second assignment it decreases sharply from 100% to 54.55%. But the assignments' similarity rates of more than 77% of our students were high because we could only detect very small portion of similarities. For the 2013 Fall semester, there is no statistically significant difference between plagiarism behavior of students for assignments 1 and 2 ($p = 0.858 > 0.05$); for assignments 1 and 3 ($p = 0.060 > 0.05$). But there is a statistically significiant difference between the students' behavior for the assignments 1 and 4 ($p = 0.006 < 0.05$); for assignments 1 and 5 ($p = 0.001 < 0.05$). There is no statistically significiant difference between the students' behavior for assignments 2 and 3 ($p = 0.146 >$

0.05); but for assignments 2 and 4 ($p = 0.006 < 0.05$) and for assignments 2 and 5 ($p = 0.000 < 0.05$), there is a significiant difference. There is no statistically significiant difference between the students' behavior for assignments 3 and 4 ($p = 0.229 > 0.05$); but for assignments 3 and 5 ($p = 0.056 > 0.05$), and for assignments 4 and 5 ($p = 0.010 < 0.05$), there is a siginificiant difference.

Table 4.5. Students' Plagiarism Behavior for Assignments in 2013 Fall Semester

Similarity Rate	**A1**	**A2**	**A3**	**A4**	**A5**
0-50%	2.17 %	0 %	6.67 %	11.54 %	22.73 %
51-70%	6.52 %	0 %	20 %	7.69 %	22.73 %
71-100%	91.30 %	100 %	73.33 %	80.77 %	54.55%

A: Assignment

As it can be easily seen from Table 4.5, giving feedback about the plagiarism behaviour to our students decreases plagiarism rate. Since we did not use an automated plagiarism detection tool, we could not detect all students who gave similar assignments and can not reduce plagiarism as in the case for Data Structures course.

In the Data Organization and Management course both in 2012 and 2013 Fall semesters, students had five assignments, 1 midterm and 1 final exams. For each student, we computed the weighted average of the grades given to assignments and exams, and this value was taken as the final score of the student. Then, we applied the curve system to these final scores and letter grades were determined. In Table 4.6 percentage of letter grades given to our students for the Data Organization and Management course offered in 2012 Fall, and 2013 Fall semesters are listed.

As shown in Table 4.6, the percentage of our successful students increases in 2013 Fall semester. For example in 2012 Fall semester less than 2% of the students got AA from the class, while in 2013 Fall semester 8.22% of the students got AA from the same class, so the percentage of successfully passing increased by 6.79%. Also in 2012 Fall semester, 35.71% of the students failed from the class by getting FD, FF, and NA grades. However in 2013 Fall semester, 35.62% of the students failed from the class, so the percentage of failing decreased by 0.09%. We also made independent samples t-test to show the statistical difference between the grades for

the 2012 and 2013 Fall semesters. According to the t-test, there is a statistically significant difference ($p = 0.004$) difference between the grades. It is also observed that the mean of the grades was 1.49 out of 4.00 for 2012 Fall semester, and the mean value increased to 1.66 out of 4.00 in 2013 Fall semester.

As shown in Table 4.3, when a plagiarism detection tool was used, the 78% of students passed the class. However, according to the results of the Table 4.6, the rate of the passing class decreased to 64%, because we detected students with a high similarity rate manually and that's why we could only detect very small portion of students who gave similar assignments. Therefore, we could not enforce our students to work harder to do their assignments. So, the rate of passing students was decreased. It was observed that no plagiarism detection tool usage affected negatively the success rate of students. Therefore, after these results, we decided to use a plagiarism detection tool embedded in Moodle system.

Table 4.6. Students' Performances for the Data Structures Course in 2012 and 2013 Fall Semesters

Letter Grade	Weight of Letter Grade (Out of 4.00)	% of Students for 2012 Fall Semester	% of Students for 2013 Fall Semester
AA	4.00	1.43	8.22
BA	3.50	5.00	4.11
BB	3.00	5.00	6.85
CB	2.50	15.71	16.44
CC	2.00	20.00	12.33
DC	1.50	8.57	9.59
DD	1.00	8.57	6.85
FD	0.50	20.71	23.29
FF	0	9.29	8.22
NA	-	5.71	4.11
TOTAL		100	100

4.2. Experiments for Determining the Distance Metrics

To determine the distance metric to use in this study, we compared some distance metrics including Jaro, Levenshtein, EuclideanDistance, JaccardSimilarity, QGramsDistance, CosineSimilarity, JaroWinkler, BlockDistance, ChapmanLengthDeviation, ChapmanMatchingSoundex, ChapmanMeanLength, ChapmanOrderedNameCompoundSimilarity, MatchingCoefficient, MongeElkan,

NeedlemanWunch, OverlapCoefficient, SmithWaterman, SmithWatermanGotoh, SmithWatermanGotohWindowedAffine, Soundex, [TagLink_[similaritymetrics.TagLinkToken@305f6ed4], [TagLinkToken_Tr_0.3] from SimMetrics Similarity Metric Library[7].

In all experiments, the successful string matching algorithms were determined and marked with a "*" character. The unsuccessful algorithms were marked with a "x" character and the average algorithms were marked with a "+" character in the tables. The value 1.0 means in Table 4.7, 4.8, 4.9, 4.10, and 4.11, the texts are identical, while 0.0 means they are not similar.

For the first experiment, our aim is to find the successful metrics when the pattern and the string are the same, partially same or completely different. The pattern and the string were selected as follows:

Pattern: gcatcgcagagagtatacagtacg

String: gcatcgcagagagtatacagtacg

When the pattern and the string were the same, the results of distance metrics are showed in Table 4.7. According to the Table 4.7, all algorithms are successful to detect the exact copy in the text.

[7] http://sourceforge.net/projects/simmetrics/

Table 4.7. The results of the first experiment with the distance metrics

Distance Metrics	**Metric**
Jaro(*)	1.0
Levenshtein(*)	1.0
EuclideanDistance(*)	1.0
JaccardSimilarity(*)	1.0
QGramsDistance(*)	1.0
CosineSimilarity(*)	1.0
JaroWinkler(*)	1.0
BlockDistance(*)	1.0
ChapmanLengthDeviation(*)	1.0
ChapmanMatchingSoundex(*)	1.0
ChapmanMeanLength(*)	1.0
ChapmanOrderedNameCompoundSimilarity(*)	1.0
MatchingCoefficient(*)	1.0
MongeElkan(*)	1.0
NeedlemanWunch(*)	1.0
OverlapCoefficient(*)	1.0
SmithWaterman(*)	1.0
SmithWatermanGotoh(*)	1.0
SmithWatermanGotohWindowedAffine(*)	1.0
Soundex(*)	1.0
[TagLink_[similaritymetrics.TagLinkToken@305f6ed4] (*)	1.0
[TagLinkToken_Tr_0.3] (*)	1.0

* successful, + average, x unsuccessful

When the string contains the pattern, the results of distance metrics are showed in Table 4.8. According to the Table 4.8, Jaro, Jaro Winkler, and Chapman Ordered Name Compound Similarity are more successful to detect the copy in the text.

Pattern: gcagagag

String: gcatcgcagagagtatacagtacg

Table 4.8. The results of the second experiment with the distance metrics

Distance Metrics	**Metric**
Jaro(*)	0.6666667
Levenshtein	0.3333333
EuclideanDistance(+)	0.0
JaccardSimilarity	0.0
QGramsDistance(+)	0.5
CosineSimilarity	0.0
JaroWinkler(*)	0.7666667
BlockDistance	0.0
ChapmanLengthDeviation	0.0
ChapmanMatchingSoundex(+)	0.53333336
ChapmanMeanLength	0.23245585
ChapmanOrderedNameCompoundSimilarity(*)	0.76666665
MatchingCoefficient	0.0
MongeElkan(+)	1.0
NeedlemanWunch(+)	0.5833334
OverlapCoefficient	0.0
SmithWaterman(+)	1.0
SmithWatermanGotoh(+)	1.0
SmithWatermanGotohWindowedAffine(+)	1.0
Soundex(+)	0.53333336
[TagLink_[similaritymetrics.TagLinkToken@305f6ed4] (+)	0.53812504
[TagLinkToken_Tr_0.3] (+)	0.615

* successful, + average, x unsuccessful

When the string does not contain the pattern, then the expected result was 0, because the pattern and the string were completely different. The results of distance metrics are showed in Table 4.9.

Pattern: u

String: gcatcgcagagagtatacagtacg

According to the Table 4.9, Jaro, Levenshtein, Euclidean Distance, Jaccard Similarity, Q-Grams Distance, Cosine Similarity, Jaro Winkler, Block Distance, Chapman Length Deviation, Matching Coefficient, OverlapCoefficient, and SmithWaterman are successful to detect this case.

Table 4.9. The results of the third experiment with the distance metrics

Distance Metrics	**Metric**
Jaro(*)	0.0
Levenshtein(*)	0.0
EuclideanDistance(*)	0.0
JaccardSimilarity(*)	0.0
QGramsDistance(*)	0.0
CosineSimilarity(*)	0.0
JaroWinkler(*)	0.0
BlockDistance(*)	0.0
ChapmanLengthDeviation(*)	0.0
ChapmanMatchingSoundex	0.33333334
ChapmanMeanLength	0.18549377
ChapmanOrderedNameCompoundSimilarity	0.16666667
MatchingCoefficient(*)	0.0
MongeElkan	0.6
NeedlemanWunch	0.5
OverlapCoefficient(*)	0.0
SmithWaterman(*)	0.0
SmithWatermanGotoh	0.6
SmithWatermanGotohWindowedAffine	0.6
Soundex	0.33333334
[TagLink_[similaritymetrics.TagLinkToken@305f6ed4](+)	0.09009199
[TagLinkToken_Tr_0.3]	0.12840697

* successful, + average, x unsuccessful

In the first experiment, the pattern and the string were consisted of one word. According to these results, Jaro, Jaro-Winkler, Cosine Similarity, Euclidean Distance, Levenshtein, Block Distance and Q-Grams distance metrics can be considered successful.

For the second experiment, all distance metrics were compared with one another using with SimMetrics Similarity Metric Library, SecondString Project[8] and Document Similarity Tool[9] of YTÜ Kemik Group for longer text documents. To test the distance metrics, a dataset having three texts was created from the YTÜ Kemik web site. These texts are given in Table 4.10 in Appendices. These texts were selected because these three texts are somewhat similar to each other and their lengths are different. So the expected results are not too high. The results should vary between 25% and 60% for each comparison. The results of this experiment are

[8] http://secondstring.sourceforge.net/
[9] http://www.kemik.yildiz.edu.tr/data/DocumentSimilarity.rar

showed in Table 4.11. The value 1.0 means in Table 4.11, the texts are identical, while 0.0 means they are not similar.

Table 4.11. The results of the experiments of Section 4.2

Distance Metrics	1&2 Similarity	1&3 Similarity	2&3 Similarity
Jaro(x)	0.0	0.0	0.0
Levenshtein(*)	0.26	0.11	0.3
EuclideanDistance(x)	0.90	0.93	0.17
JaccardSimilarity(+)	0.12	0.06	0.09
QGramsDistance(*)	0.35	0.19	0.3
CosineSimilarity(*)	0.44	0.45	0.56
JaroWinkler(x)	0.0	0.0	0.0
Dice(+)	0.19	0.14	0.2
BlockDistance(+)	0.17	0.09	0.14
ChapmanLengthDeviation(x)	0.0	0.0	0.0
ChapmanMatchingSoundex(x)	0.72	0.84	0.79
ChapmanMeanLength(x)	1.0	1.0	1.0
ChapmanOrderedNameCompoundSimilarity(+)	0.24	0.27	0.25
MatchingCoefficient(x)	0.14	0.05	0.09
MongeElkan(x)	0.91	0.94	0.92
NeedlemanWunch(+)	0.54	0.51	0.52
OverlapCoefficient(x)	0.0	0.0	0.0
SmithWaterman(x)	0.05	0.065	0.055
SmithWatermanGotoh(+)	0.14	0.20	0.15
SmithWatermanGotohWindowedAffine(x)	0.06	0.05	0.03
Soundex(x)	0.0	0.33	0.0
[TagLink_[similaritymetrics.TagLinkToken@305f6ed4](+)	0.50	0.48	0.51
[TagLinkToken_Tr_0.3](+)	0.14	0.1	0.11

* successful, + average, x unsuccessful

According to the results presented in Table 4.11, Cosine Similarity, and Q-Grams distance metrics can be considered most successful metrics. Block Distance, JaccardSimilarity, Dice, Levenshtein, Needleman Wunch, Smith Waterman Gotoh, [TagLink_[similaritymetrics.TagLinkToken@305f6ed4], Chapman Ordered Name Compound Similarity, and [TagLinkToken_Tr_0.3] can be considered successful, too. Some others like Jaro produced too low similarity, somemetrics like Chapman Mean Length computed too high similarity. After these results, we decided to use the Cosine Similarity as a distance metric because of its success.

4.3. Experiments for Determining the Distance Metrics After The Preprocessing Phases

For data preprocessing, Turkish characters were removed. All letters were transformed into lowercase. For the similarity metrics, the stemming process and the removal of Turkish stopwords were compared in this thesis. Also in this experiment we studied the YTÜ Kemik Dataset from the second experiment in Section 4.2 because the similarity metrics were not limited to a certain number of character comparisons. The distance metrics explained in Literature Review and tested in the experiments in Section 4.2 run with SimMetrics Similarity Metric Library, SecondString Project and Document Similarity Tool of YTÜ Kemik.

For this experiment, after the preprocessing, the texts were compared with one another again. The comparisons of these results are showed in Table 4.12 in Appendices.

According to these results, the preproccesing phase increases the performance of plagiarism detection. Especially, using Zemberek increases the performance. According to Table 4.12, one of the successful metrics, Q-Grams, was more successful when stopwords and Turkish characters were removed, but the succesful similarity metrics, CosineSimilarity, NeedlemanWunch, and [TagLink_[similaritymetrics.TagLinkToken@305f6ed4] gave the best results when the stopwords and Turkish characters were removed and the stemming process using Zemberek was done. Among them, the Cosine Similarity is more successful.

After these results, we decided to use the Cosine Similarity as a distance metric because of its success. In addition, from the Previous Work section we observed that the Cosine Similarity is commonly used in text mining. This is also another reason why we choose the Cosine Similarity as a distance metric in this thesis.

4.4. Experiment for Comparing Cosine Similarity with Other Well-Known Metrics

In this experiment, Cosine Similarity were compared with other well-known distance metrics which are Dice, Jaccard, and Euclidean metrics by using a dataset collected from the text documents submitted by the students for the homeworks of the Data Management and Organization Course of our Moodle web site at the Çukurova University Computer Engineering Department. These texts are given in Table 4.13 in Appendices.

For data preprocessing, firstly all letters were transformed into lowercase, then the stemming and numeric character removal were applied. For stemming, Zemberek was used. Four similarity metrics including Euclidean Similarity, Jaccard Similarity, Dice Similarity and Cosine Similarity were compared with one another. The first and the second dataset were compared with one another in different ways. The preprocessing methods are given in Table 4.14. The results of the experiment are showed in Table 4.15.

Table 4.14. The Preprocessing Methods

ID	Stopword Removal	Stemming	Numeric Character Removal	Turkish Character Removal	Lowercase
1	No	No	No	No	Yes
2	No	No	Yes	No	Yes
3	Yes	No	Yes	No	Yes
4	No	Yes	Yes	No	Yes
5	Yes	Yes	Yes	No	Yes
6	No	No	No	Yes	Yes
7	No	No	Yes	Yes	Yes
8	Yes	No	Yes	Yes	Yes
9	No	Yes	Yes	Yes	Yes
10	Yes	Yes	Yes	Yes	Yes

Table 4.15. The Similarity Values for Each Distance Metric

Preprocessing	Euclidean	Jaccard	Dice	Cosine
1	7.5%	44.77%	61.85%	61.85%
2	8.8%	47.32%	64.24%	64.31%
3	8.8%	47.32%	64.24%	64.31%
4	6.8%	52.42%	68.78%	69.03%
5	8.1%	58.03%	73.44%	73.45%
6	7.5%	34.63%	51.45%	51.54%
7	7.6%	46.55%	63.52%	63.52%
8	9%	49.75%	66.45%	66.54%
9	8%	58.26%	73.62%	73.62%
10	8.2%	58.80%	74.06%	**74.10%**

According to these results the best similarity metric is Cosine Similarity for these experiments. Also, this table shows that preprocessing is more important to find similarities between the textual documents.

4.5. A Comparison of Greedy String Tiling Algorithm with Cosine Similarity

Although the Cosine Similarity metric is the best performer according to the above experimental results, the Greedy Sting Tiling Algorithm is selected as a base algorithm for detecting similarities instead of the Cosine Similarity Algorithm, because the Cosine Similarity Algorithm is sensitive to word frequency. If some words are frequently used in the text, these higher frequency words will tend to get higher similarity scores. That's why, the Cosine Similarity is only good to identify near identical documents. The other reason of selecting the Greedy String Tiling Algorithm is that the order of the words in the text becomes lost in the Cosine Similarity; however their ordering is important for determining plagiarism.

To show the comparison of GST with Cosine Similarity, we used two experiments. Firstly, we prepared texts in Table 4.16, which contain common words, but basically are not identical. The similarity scores of the texts 1 and 2 in Table 4.16 are presented in Table 4.17.

Table 4.16. The compared datasets

Text1	Text2
Bir bilgisayar satın alırken İşlemci Bellek ve Sabit disk özelliklerine mutlaka dikkat edilmelidir. İşlemci bilgisayar alırken dikkat gerektiren en önemli parçadır. Bu sırayı Bellek ve Sabit disk takip etmektedir. Bu yüzden bilgisayar parçalarının özelliklerini mutlaka araştırınız.	İşlemci bellek aracılığıyla veri alırken yaptığı hesaplama sonucunun mutlaka bilgisayar üzerinde bulunan Sabit disk içerisine kaydedilmiş olmasına dikkat edilmelidir. Bu çok önelidir. Bu yüzden Sabit disk boyutuna dikkat edilmelidir.

Table 4.17. Similarity Scores Computed by GST and Cosine Similarity

Files	Cosine	GST (m=1)	GST (m=2)
Text1-Text2	57.88	47.22	27.78

According to the Table 4.17, when the Cosine Similarity, and GST Algorithm with minimum match length as taken 1 were used, it was observed that these texts were similar. Because the Cosine Similarity and our proposed system with minimum match length as taken 1 are sensitive to word frequency, but they are not sensitive to word order. On the other hand, the GST algorithm with minimum match length as taken 2 is sensitive to word order. According to its result, these texts are not similar. That's why; to obtain the correct result, we must use an algorithm which is sensitive to word order. In that case, GST is more selectable algorithm than the Cosine Similarity.

Secondly, we used some assignments as a different dataset to compare GST with Cosine Similarity. The used assignments in this experiment are not similar to each other. The assignments are shown in Table 4.18 in Appendices and the results are shown in Table 4.19.

Table 4.19. Similarity Scores Computed by GST and Cosine Similarity for homeworks

First Document	Second Document	Cosine	GST (m=1)
File_1	File_2	52.65	17.71
File_2	File_3	52.33	17.27
File_2	File_4	56.58	22.37
File_2	File_5	40.2	9.42
File_2	File_6	42.05	10.97

4.6. Experiments with Greedy String Tiling Algorithm

First of all, the tokenization process was completed. Then our string matching algorithm was applied to the same dataset in Section 4.4. The preprocessing methods are given in Table 4.14.

The minimum match length was taken as 1 and 2. The results of the preprocessing for GST are showed in Table 4.20.

Table 4.20. The results and the comparisons of the preprocessing for GST

Preprocessing	**GST (m=1)**	**GST (m=2)**
1	38.71%	19.35%
2	44.90%	27.55%
3	48.41%	26.75%
4	52.31%	26.67%
5	58.97%(*)	44.87%(*)
6	41.94%	19.35%
7	46.94%	30.61%
8	50.96%	30.57%
9	53.61%	30.93%
10	58.97%(*)	44.87%(*)

* successful

According to these results, the best methods are using preprocessing, stopword and stemming phases while minimum match length was taken as 1 or 2. The results are similar to the results in Section 4.4.

4.7. Effect of Using Dictionaries

In this experiment, the comparison of our proposed GST Algorithm with preprocessing phases including lowercasing, numeric character removal, stemming, and using dictionaries and without preprocessing phases are done. The compared datasets are shown as in Table 4.21. The texts are basically identical, only in one of the assignment the abbreviation and synonym of the words are used. The results of this experiment are showed in Table 4.22. The best results are written in bold face in Table 4.22.

Table 4.21. The compared datasets for preprocessing

Text1	Text2
32 MB Random Access Memory ile çalışan hardware 64 MB olan donanıma göre daha yavaştır. Bu karşılaştırmada CPU nitelikleri aynıdır.	128 MegaByte RAM ile çalışan donanım 256 megabayt olan donanıma göre daha da ağırdır. Bu karşılaştırmada merkezi işlem birimi niteliği eştir.

Table 4.22. The comparisons of GST (m=1), GST (m=2), and GST (m=1 & no preprocess)

	GST (m=1)	GST (m=2)	GST (m=1 & no preprocess)
Text1-Text2	**100**	**100**	45.45
Text2-Text1	**100**	**100**	43.48

As shown in Table 4.22, stopword removal, using dictionary of synonyms, using abbreviation list, and using glossary of computer terms are important for plagiarism detection, and these features improve similarity score computations.

4.8. Comparison of the Proposed System with the Existing Tools

In this experiment, the selected datasets were compared with SIM, WCopyfind, Sherlock, CopyCatchGold, JPlag, Ferret, Text Matching System and Document Similarity, two other software developed by YTÜ Kemik Group, as well as our implemented Cosine, Dice, Euclidean, and Jaccard text similarity measures and our proposed GST Algorithm prepared in this study. In addition, we investigated the affects of using a Turkish character set and a Turkish stemmer on plagiarism detection. In this experiment, we used some variations of texts given in Table 4.23 and 4.24. The first and second texts presented in Table 4.23 are less similar to each other, but they have common words, while the third and fourth texts in Table 4.24 are very similar to each other.

Two short texts taken from the first page and back cover page of the book "Transformation" from the Franz Kafka (1915) are shown in Table 4.23 constituted the first data set. Then, Turkish characters in these two texts were converted to Latin characters and this version of texts formed the second data set. The third data set was created by stemming of Turkish words in the first dataset (Kaya and Özel, 2014a). The process of the word stemming was performed with the Zemberek software. The

reason of using Zemberek was that Zemberek produced better results when compared to the other stemming methods and is a preferable stemming method for Turkish text applications due to high dimensionality reduction rate it provides (Tunalı & Bilgin, 2012). In the fourth data set, the processes in the second and third data set are combined. After the character transformation process, the roots of some words could not be found. These roots were corrected by the "Kelime Öner" method of Zemberek, which generates suggestions for the word, and then analyzed again (Kaya and Özel, 2014b).

Table 4.23. Text1 and Text2

Text1.txt	**Text2.txt**
"Gregor Samsa bir sabah huzursuz düşlerinden uyandığında, kendini yatağında dev bir böceğe dönüşmüş olarak buldu. Zırhı andıran sertlikteki sırtının üzerinde yatmaktaydı ve başını azıcık yukarı kaldırdığında kubbemsi, kahverengi, yay biçimindeki sertliklerce bölümlenmiş; üstünde, tutunabileceği hiçbir şey kalmamış ve neredeyse kaymak üzere olan yorganın bulunduğu karnını gördü..." - Franz Kafka, Dönüşüm.	"Gregor Samsa bir sabah kötü bir rüyadan uyandiginda, kendini yataginda korkunç bir böceğe dönüşmüş olarak buldu. Yaşadığı cagin zihniyetine o kadar büyük bir tepki duyar ki Kafka, ölümünden sonra yazdığı bütün eserlerini yakılması için en yakın dostu Max Brody'ye emanet eder. Bugün Kafka gibi büyük bir edebi dehayı okuyabiliyorsak eğer, bunu Max Brody'nin ihanetine borçluyuz..."

Two short texts taken from the different film web sites from the “Step Up All In”[10,11] film summaries as shown in Table 4.24 constituted the fifth dataset. The processing applied to the second, third and fourth dataset were applied to this new dataset in the same order, creating the sixth, seventh and eighth datasets (Kaya and Özel, 2014b).

We used these datasets to compare the performance of our system in text files which are not computer science related documents, to show the general behavior of our system.

Table 4.24. Text3 and Text4

[10] http://www.beyazperde.com/
[11] http://www.hdfilmvadisi.com/

Text3.txt	Text4.txt
Sean için hayat hiç de planladığı gibi gitmemektedir. Kızarkadaşı kariyerinde farklı bir şekilde devam etme kararı almıştır. Kurduğu grup olan The Mob, Los Angeles'taki yarısı kazanamayınca Sean'ı arkalarında bırakıp geri dönerler. Dansçı olarak kariyerini sürdürmek isteyen Sean, eski arkadaşı Moose ile yeni bir ekip aramaya başlarlar. Moose daha sonra Sean'ı, sakatlığı nedeniyle bir süredir dans etmeyen fakat sahneye dönmeye artık hazır olan Andie West ile tanıştırır. Andie de Jenny Kido, Vladd ve Santiago ikizlerini toplayınca ekipleri kurulmuş olur. Yeni gurubun adı LMNTRX olur. Kariyerleri için önemli bir fırsat ve dönüm noktası olan Las Vegas'taki The Vortex yarışmasına katılmaya karar verirler. Sean'ın eski ekibi The Mob'un da yarışmada yer aldıklarını öğrendiklerinde Sean için durum biraz daha zorlaşır.	Sean için yaşam hiç de düşündüğü şeklinde gitmemektedir. Kızarkadaşı kariyerinde ayrımlı bir halde devam etme sonucu almistir. Kurduğu öbek olan The Mob, Los Angeles'taki yarısı kazanamayınca Sean'ı arkalarında bırakıp geri dönerler. Dansci olarak kariyerini sürdürmek isteyen Sean, eski arkadasi Moose ile yeni bir takım aramaya başlarlar. Moose sonrasında Sean'ı, sakatlığı sebebiyle bir süredir dans etmeyen ama sahneye dönmeye artık hazır olan Andie West ile tanıştırır. Andie de Jenny Kido, Vladd ve Santiago ikizlerini toplayınca ekipleri kurulmuş olur. Yeni gurubun adı LMNTRX olur. Kariyerleri için mühim bir fırsat ve dönüm noktası olan Las Vegas'taki The Vortex yarışmasına katılmaya karar verirler. Sean'ın eski kadrosu The Mob'un da yarışmada yer aldıklarını öğrendiklerinde Sean için vaziyet birazcık daha fazla zorlaşır.

The comparison results of our proposed system with the existing tools over the eight dataset are showed in Table 4.25-4.32. *T1* represents the first text; *T2* represents the second text; *T3* represents the third text; *T4* represents the fourth text; *m* represents the minimum match length; *(T)* represents that the Turkish characters were converted to the Latin characters; and *(S)* represents that stemming was done. The best results are written in boldface in the tables.

Table 4.25. The comparison result of the first dataset (% Similarity)

Method	**T1-T2**	**T2-T1**	**T1-T1**
CopyCatch Gold (m=5 character)	**26**	-	**100**
WCopyfind (m=1 word)	22	20	**100**
WCopyfind (m=2 word)	18	16	**100**
WCopyfind (m=3 word)	18	16	**100**
WCopyfind (m=5 word)	10	9	**100**
Sim (m=1 word)	22	26	**100**
Sim (m=2 word)	18	15	**100**
Sim (m=3 word)	18	15	**100**
Sim (m=5 word)	10	8	**100**
Sherlock (m=1 word)	10	-	**100**
Sherlock (m=2 word)	11	-	**100**
Sherlock (m=3 word)	6	-	**100**
Sherlock (m=5 word)	0	-	**100**
JPlag (m=1 word)	-	-	-
JPlag (m=2 word)	16	14	97.9
JPlag (m=3 word)	16	14	97.9
JPlag (m=5 word)	10	8	97.9
Text_Matching System	17	-	**100**
Document Similarity – Cosine (m=1 word)	**31.30**	-	**100**
Document Similarity - Jaccard (m=1 word)	12.35	-	**100**
Document Similarity - Dice (m=1 word)	21.98	-	**100**
Document Similarity - Tanimoto (m=1 word)	18.33	-	**100**
Document Similarity - Cosine (m=2 word)	13.50	-	**100**
Document Similarity - Jaccard (m=2 word)	7.45	-	**100**
Document Similarity - Dice (m=2 word)	13.86	-	**100**
Document Similarity - Tanimoto (m=2 word)	7.22	-	**100**
Cosine Similarity	**29.09**	-	**100**
Jaccard Similarity	16.67	-	**100**
Dice Similarity	28.57	-	**100**
Ferret	2.56	-	**100**
DOCCop (m=8 word)	0	-	99
Turnitin	100	28	**100**
Our Proposal (m=1)	**24.39**	**32.84**	**100**
Our Proposal (m=2)	19.51	20	**100**
Our Proposal (m=3)	19.51	20	**100**
Our Proposal (m=5)	0	0	**100**

Table 4.26. The comparison result of the second dataset (% Similarity)

Method	**T1(T)-T2(T)**	**T2(T)-T1(T)**	**T1(T)-T1(T)**
CopyCatch Gold (m=5 character)	**28**	-	**100**
WCopyfind (m=1 word)	**26**	**21**	**100**
WCopyfind (m=2 word)	**26**	**21**	**100**
WCopyfind (m=3 word)	**26**	**21**	**100**
WCopyfind (m=5 word)	10	9	**100**
Sim (m=1 word)	**26**	**27**	**100**
Sim (m=2 word)	**26**	**27**	**100**
Sim (m=3 word)	**26**	**21**	**100**
Sim (m=5 word)	10	9	**100**
Sherlock (m=1 word)	15	-	**100**
Sherlock (m=2 word)	17	-	**100**
Sherlock (m=3 word)	6	-	**100**
Sherlock (m=5 word)	0	-	**100**
JPlag (m=1 word)	-	-	-
JPlag (m=2 word)	**23**	**20**	97.8
JPlag (m=3 word)	**23**	**20**	97.8
JPlag (m=5 word)	10	9	97.8
Text_Matching System	18	-	**100**
Document Similarity - Cosine (m=1 word)	**31.75**	-	**100**
Document Similarity - Jaccard (m=1 word)	13.92	-	**100**
Document Similarity - Dice (m=1 word)	24.44	-	**100**
Document Similarity - Tanimoto (m=1 word)	18.42	-	**100**
Document Similarity - Cosine (m=2 word)	17.93	-	**100**
Document Similarity - Jaccard (m=2 word)	10.11	-	**100**
Document Similarity - Dice (m=2 word)	18.37	-	**100**
Document Similarity - Tanimoto (m=2 word)	9.78	-	**100**
Cosine Similarity	**31.43**	-	**100**
Jaccard Similarity	18.02	-	**100**
Dice Similarity	**30.53**	-	**100**
Ferret	6.59	-	**100**
DOCCop (m=8 word)	0	-	99
Turnitin	100	28	**100**
Our Proposal (m=1)	**26.32**	**25**	**100**
Our Proposal (m=2)	**26.32**	**25**	**100**
Our Proposal (m=3)	21.05	20	**100**
Our Proposal (m=5)	0	0	**100**

Table 4.27. The comparison result of the third dataset (% Similarity)

Method	**T1(S)-T2(S)**	**T2(S)-T1(S)**	**T1(S)-T1(S)**
CopyCatch Gold (m=5 character)	**30**	-	**100**
WCopyfind (m=1 word)	**28**	**25**	**100**
WCopyfind (m=2 word)	24	21	**100**
WCopyfind (m=3 word)	24	21	**100**
WCopyfind (m=5 word)	10	9	**100**
Sim (m=1 word)	**30**	**30**	**100**
Sim (m=2 word)	24	21	**100**
Sim (m=3 word)	24	21	**100**
Sim (m=5 word)	10	9	**100**
Sherlock (m=1 word)	7	-	**100**
Sherlock (m=2 word)	11	-	**100**
Sherlock (m=3 word)	0	-	**100**
Sherlock (m=5 word)	0	-	**100**
JPlag (m=1 word)	-	-	-
JPlag (m=2 word)	21	20	98.5
JPlag (m=3 word)	21	20	98.5
JPlag (m=5 word)	11	10	98.5
Text_Matching System	25	-	**100**
Document Similarity – Cosine (m=1 word)	**34.72**	-	**100**
Document Similarity – Jaccard (m=1 word)	15	-	**100**
Document Similarity – Dice (m=1 word)	26.09	-	**100**
Document Similarity – Tanimoto (m=1 word)	20.69	-	**100**
Document Similarity – Cosine (m=2 word)	17.36	-	**100**
Document Similarity – Jaccard (m=2 word)	9.78	-	**100**
Document Similarity – Dice (m=2 word)	17.82	-	**100**
Document Similarity – Tanimoto (m=2 word)	9.47	-	**100**
Cosine Similarity	**34.72**	-	**100**
Jaccard Similarity	20.69	-	**100**
Dice Similarity	**34.29**	-	**100**
Ferret	6.38	-	**100**
DOCCop (m=8 word)	0	-	99
Turnitin	0	0	**100**
Our Proposal (m=1)	**27.50**	**28.21**	**100**
Our Proposal (m=2)	22.50	23.08	**100**
Our Proposal (m=3)	17.50	17.95	**100**
Our Proposal (m=5)	0	0	**100**

Table 4.28. The comparison result of the fourth dataset (% Similarity)

Method	**T1(T,S)-T2(T,S)**	**T2(T,S)-T1(T,S)**	**T1(T,S)-T1(T,S)**
CopyCatch Gold (m=5 character)	**28**	-	**100**
WCopyfind (m=1 word)	**28**	**23**	**100**
WCopyfind (m=2 word)	26	21	**100**
WCopyfind (m=3 word)	26	21	**100**
WCopyfind (m=5 word)	0	0	**100**
Sim (m=1 word)	**28**	**29**	**100**
Sim (m=2 word)	26	21	**100**
Sim (m=3 word)	26	21	**100**
Sim (m=5 word)	10	9	**100**
Sherlock (m=1 word)	10	-	**100**
Sherlock (m=2 word)	11	-	**100**
Sherlock (m=3 word)	0	-	**100**
Sherlock (m=5 word)	0	-	**100**
JPlag (m=1 word)	-	-	-
JPlag (m=2 word)	22	22	98.3
JPlag (m=3 word)	22	22	98.3
JPlag (m=5 word)	12	12	98.3
Text_Matching System	12	-	**100**
Document Similarity - Cosine (m=1 word)	**32.81**	-	**100**
Document Similarity - Jaccard (m=1 word)	14.67	-	**100**
Document Similarity - Dice (m=1 word)	25.58	-	**100**
Document Similarity - Tanimoto (m=1 word)	18.85	-	**100**
Document Similarity - Cosine (m=2 word)	17.62	-	**100**
Document Similarity - Jaccard (m=2 word)	10.23	-	**100**
Document Similarity - Dice (m=2 word)	18.56	-	**100**
Document Similarity - Tanimoto (m=2 word)	9.57	-	**100**
Cosine Similarity	**32.81**	-	**100**
Jaccard Similarity	18.85	-	**100**
Dice Similarity	**31.72**	-	**100**
Ferret	6.59	-	**100**
DOCCop (m=8 word)	0	-	99
Turnitin	0	0	**100**
Our Proposal (m=1)	**28.21**	**28.21**	**100**
Our Proposal (m=2)	23.08	23.08	**100**
Our Proposal (m=3)	17.95	17.95	**100**
Our Proposal (m=5)	0	0	**100**

Table 4.29. The comparison result of the fifth dataset (% Similarity)

Method	**T3-T4**	**T4-T3**	**T3-T3**
CopyCatch Gold (m=5 character)	**83**	-	**100**
WCopyfind (m=1 word)	**84**	**84**	**100**
WCopyfind (m=2 word)	78	78	**100**
WCopyfind (m=3 word)	73	73	**100**
WCopyfind (m=5 word)	61	61	**100**
Sim (m=1 word)	**85**	**84**	**100**
Sim (m=2 word)	81	81	**100**
Sim (m=3 word)	76	76	**100**
Sim (m=5 word)	62	62	**100**
Sherlock (m=1 word)	66	-	**100**
Sherlock (m=2 word)	50	-	**100**
Sherlock (m=3 word)	44	-	**100**
Sherlock (m=5 word)	42	-	**100**
JPlag (m=1 word)	-	-	-
JPlag (m=2 word)	73	73	95
JPlag (m=3 word)	67	67	95
JPlag (m=5 word)	52	52	95
Text_Matching System	**86**	-	**100**
Document Similarity – Cosine (m=1 word)	**90.33**	-	**100**
Document Similarity – Jaccard (m=1 word)	68.58	-	**100**
Document Similarity – Dice (m=1 word)	81.36	-	**100**
Document Similarity – Tanimoto (m=1 word)	82.30	-	**100**
Document Similarity – Cosine (m=2 word)	70.59	-	**100**
Document Similarity – Jaccard (m=2 word)	52.70	-	**100**
Document Similarity – Dice (m=2 word)	69.03	-	**100**
Document Similarity – Tanimoto (m=2 word)	54.55	-	**100**
Cosine Similarity	**88.35**	-	**100**
Jaccard Similarity	79.12	-	**100**
Dice Similarity	**88.34**	-	**100**
Ferret	44.85	-	**100**
DOCCop (m=8 word)	66	-	99
Turnitin	**87**	**87**	**100**
Our Proposal (m=1)	**84.62**	**84.62**	**100**
Our Proposal (m=2)	**79.12**	**79.12**	**100**
Our Proposal (m=3)	74.73	74.73	**100**
Our Proposal (m=5)	67.03	67.03	**100**

Table 4.30. The comparison result of the sixth dataset (% Similarity)

Method	**T3(T)-T4(T)**	**T4(T)-T3(T)**	**T3(T)-T3(T)**
CopyCatch Gold (m=5 character)	**85**	-	**100**
WCopyfind (m=1 word)	**87**	**87**	**100**
WCopyfind (m=2 word)	81	81	**100**
WCopyfind (m=3 word)	75	75	**100**
WCopyfind (m=5 word)	66	66	**100**
Sim (m=1 word)	**87**	**86**	**100**
Sim (m=2 word)	**84**	**84**	**100**
Sim (m=3 word)	77	77	**100**
Sim (m=5 word)	67	67	**100**
Sherlock (m=1 word)	74	-	**100**
Sherlock (m=2 word)	55	-	**100**
Sherlock (m=3 word)	50	-	**100**
Sherlock (m=5 word)	75	-	**100**
JPlag (m=1 word)	-	-	-
JPlag (m=2 word)	77	77	95
JPlag (m=3 word)	69	69	95
JPlag (m=5 word)	61	61	95
Text_Matching System	**84**	-	**100**
Document Similarity - Cosine (m=1 word)	**91.68**	-	**100**
Document Similarity - Jaccard (m=1 word)	72.12	-	**100**
Document Similarity - Dice (m=1 word)	**83.80**	-	**100**
Document Similarity - Tanimoto (m=1 word)	**84.62**	-	**100**
Document Similarity - Cosine (m=2 word)	74.79	-	**100**
Document Similarity - Jaccard (m=2 word)	58.04	-	**100**
Document Similarity - Dice (m=2 word)	73.45	-	**100**
Document Similarity - Tanimoto (m=2 word)	59.73	-	**100**
Cosine Similarity	**90.19**	-	**100**
Jaccard Similarity	82.12	-	**100**
Dice Similarity	**90.18**	-	**100**
Ferret	49.07	-	**100**
DOCCop (m=8 word)	66	-	99
Turnitin	**87**	**87**	**100**
Our Proposal (m=1)	**85.06**	**85.06**	**100**
Our Proposal (m=2)	**80.46**	**80.46**	**100**
Our Proposal (m=3)	73.56	73.56	**100**
Our Proposal (m=5)	70.11	70.11	**100**

Table 4.31. The comparison result of the seventh dataset (% Similarity)

Method	**T3(S)-T4(S)**	**T4(S)-T3(S)**	**T3(S)-T3(S)**
CopyCatch Gold (m=5 character)	**88**	-	**100**
WCopyfind (m=1 word)	**88**	**88**	**100**
WCopyfind (m=2 word)	**82**	**82**	**100**
WCopyfind (m=3 word)	77	77	**100**
WCopyfind (m=5 word)	65	65	**100**
Sim (m=1 word)	**91**	**88**	**100**
Sim (m=2 word)	**86**	**86**	**100**
Sim (m=3 word)	77	77	**100**
Sim (m=5 word)	65	65	**100**
Sherlock (m=1 word)	75	-	**100**
Sherlock (m=2 word)	64	-	**100**
Sherlock (m=3 word)	50	-	**100**
Sherlock (m=5 word)	0	-	**100**
JPlag (m=1 word)	-	-	-
JPlag (m=2 word)	**87**	**83**	99
JPlag (m=3 word)	82	78	99
JPlag (m=5 word)	74	71	99
Text_Matching System	**90**	-	**100**
Document Similarity – Cosine (m=1 word)	**93.27**	-	**100**
Document Similarity – Jaccard (m=1 word)	75	-	**100**
Document Similarity – Dice (m=1 word)	**85.71**	-	**100**
Document Similarity – Tanimoto (m=1 word)	**87.34**	-	**100**
Document Similarity – Cosine (m=2 word)	77.42	-	**100**
Document Similarity – Jaccard (m=2 word)	60	-	**100**
Document Similarity – Dice (m=2 word)	75	-	**100**
Document Similarity – Tanimoto (m=2 word)	63.16	-	**100**
Cosine Similarity	**93.02**	-	**100**
Jaccard Similarity	**86.90**	-	**100**
Dice Similarity	**92.99**	-	**100**
Ferret	48.39	-	**100**
DOCCop (m=8 word)	66	-	99
Turnitin	**88**	**88**	**100**
Our Proposal (m=1)	**88.17**	**88.17**	**100**
Our Proposal (m=2)	**83.87**	**83.87**	**100**
Our Proposal (m=3)	**81.72**	**81.72**	**100**
Our Proposal (m=5)	70.97	70.97	**100**

Table 4.32. The comparison result of the eighth dataset (% Similarity)

Method	**T3(T,S)-T4(T,S)**	**T4(T,S)-T3(T,S)**	**T3(T,S)-T3(T,S)**
CopyCatch Gold (m=5 character)	**86**	-	**100**
WCopyfind (m=1 word)	**88**	**88**	**100**
WCopyfind (m=2 word)	80	80	**100**
WCopyfind (m=3 word)	77	77	**100**
WCopyfind (m=5 word)	65	65	**100**
Sim (m=1 word)	**90**	**87**	**100**
Sim (m=2 word)	84	84	**100**
Sim (m=3 word)	77	77	**100**
Sim (m=5 word)	66	66	**100**
Sherlock (m=1 word)	74	-	**100**
Sherlock (m=2 word)	68	-	**100**
Sherlock (m=3 word)	60		**100**
Sherlock (m=5 word)	11		**100**
JPlag (m=1 word)	-	-	-
JPlag (m=2 word)	78	76	95
JPlag (m=3 word)	70	68	95
JPlag (m=5 word)	63	61	95
Text_Matching System	**89**	-	**100**
Document Similarity - Cosine (m=1 word)	**93.52**	-	**100**
Document Similarity - Jaccard (m=1 word)	72.83	-	**100**
Document Similarity - Dice (m=1 word)	84.27	-	**100**
Document Similarity - Tanimoto (m=1 word)	**87.83**	-	**100**
Document Similarity - Cosine (m=2 word)	76.42	-	**100**
Document Similarity - Jaccard (m=2 word)	58.57	-	**100**
Document Similarity - Dice (m=2 word)	73.87	-	**100**
Document Similarity - Tanimoto (m=2 word)	61.84	-	**100**
Cosine Similarity	**93.52**	-	**100**
Jaccard Similarity	**87.83**	-	**100**
Dice Similarity	**93.52**	-	**100**
Ferret	48.05	-	**100**
DOCCop (m=8 word)	66	-	99
Turnitin	**87**	**87**	**100**
Our Proposal (m=1)	**86.21**	**88.24**	**100**
Our Proposal (m=2)	**80.46**	**82.35**	**100**
Our Proposal (m=3)	75.86	77.65	**100**
Our Proposal (m=5)	68,97	70.59	**100**

The most successful tools for text documents in general contexts are our proposed system, Sim, WCopyfind, CopyCatch Gold, JPlag, Text_Matching System, Document Similarity – Cosine Similarity, our implemented Cosine and Dice similarities as shown in tables of all datasets. Turnitin is succesfful for similar documents as shown in tables 4.29-4.32, but for different documents Turnitin is not successful as shown in tables 4.25-4.28. The most unsuccessful tools are Sherlock, Ferret and DOCCop as shown in tables 4.25-4.32.

Also, according to tables, it was observed that using Turkish characters decreased similarity detection, while using stemmed words increased the performance of plagiarism detection tools. As seen from the results in tables, the transformation of Turkish characters to Latin characters improved the similarity calculation, while using the stemmed words further increased the similarity calculation in a positive direction, providing more successful and realistic results. However, when the Turkish characters were converted into Latin characters, and then words were stemmed, the similarity rate decreased with respect to using only stemming words. It was concluded that using stemmed words had a big effect on similarity detection of textual documents.

After receiving these results, we decided to use word stemming with Turkish characters as preprocessing. After this decision, these tools were compared to determine the most successful plagiarism detection tools with each other using the texts in Section 4.5 and 4.7 which are computer science related short texts including synonyms, abbreviations, and numeric characters. The results are given in Table 4.33 and 4.34, respectively.

Table 4.33. The comparison result of the texts in 4.5 (% Similarity)

Method	**T1-T2**	**T2-T1**	**T1-T1**
CopyCatch Gold (m=5 character)	56	-	**100**
WCopyfind (m=1 word)	55	68	**100**
WCopyfind (m=2 word)	**27**	**34**	**100**
Sim (m=1 word)	52	51	**100**
Sim (m=2 word)	**22**	**34**	**100**
Sherlock (m=1 word)	**26**	-	**100**
Sherlock (m=2 word)	12	-	**100**
JPlag (m=1 word)	-	-	-
JPlag (m=2 word)	33	25	97
Text_Matching System	59	-	**100**
Document Similarity - Cosine (m=1 word)	65.08	-	**100**
Document Similarity - Jaccard (m=1 word)	**29.73**	-	**100**
Document Similarity - Dice (m=1 word)	45.83	-	**100**
Document Similarity - Tanimoto (m=1 word)	39.13	-	**100**
Document Similarity - Cosine (m=2 word)	**24.85**	-	**100**
Document Similarity - Jaccard (m=2 word)	9.43	-	**100**
Document Similarity - Dice (m=2 word)	17.24	-	**100**
Document Similarity - Tanimoto (m=2 word)	14.06	-	**100**
Cosine Similarity	46.38	-	**100**
Jaccard Similarity	39.44	-	**100**
Dice Similarity	56.57	-	**100**
Ferret	0	-	**100**
DOCCop (m=8 word)	0	-	**100**
Turnitin	0	0	**100**
Our Proposal (m=1)	44	57.89	**100**
Our Proposal (m=2)	**24**	**31.58**	**100**

The files are not actually very similar to each other. According to the results in Table 4.33, the most successful plagiarism detection tools are our proposed system, SIM, Sherlock, WCopyfind and Document Similarity tool of YTÜ.

Table 4.34. The comparison result of the texts in Section 4.7 (% Similarity)

Method	**T1-T2**	**T2-T1**	**T1-T1**
CopyCatch Gold (m=5 character)	41	-	**100**
WCopyfind (m=1 word)	44	42	**100**
WCopyfind (m=2 word)	44	42	**100**
Sim (m=1 word)	40	38	**100**
Sim (m=2 word)	40	38	**100**
Sherlock (m=1 word)	20	-	**100**
Sherlock (m=2 word)	14	-	**100**
JPlag (m=1 word)	-	-	-
JPlag (m=2 word)	40	38	95
Text_Matching System	66	-	**100**
Document Similarity - Cosine (m=1 word)	48.80	-	**100**
Document Similarity - Jaccard (m=1 word)	35.62	-	**100**
Document Similarity - Dice (m=1 word)	51.43	-	**100**
Document Similarity - Tanimoto (m=1 word)	32.26	-	**100**
Document Similarity - Cosine (m=2 word)	28.58	-	**100**
Document Similarity - Jaccard (m=2 word)	16.67	-	**100**
Document Similarity - Dice (m=2 word)	28.57	-	**100**
Document Similarity - Tanimoto (m=2 word)	16.67	-	**100**
Cosine Similarity	37.22	-	**100**
Jaccard Similarity	22.86	-	**100**
Dice Similarity	37.21	-	**100**
Ferret	6.06	-	**100**
DOCCop (m=8 word)	0	-	**100**
Turnitin	0	0	**100**
Our Proposal (m=1)	**100**	**100**	**100**
Our Proposal (m=2)	**100**	**100**	**100**

The files are actually identical. According to the results in Table 4.34, the most successful plagiarism detection tool is our proposed system. Because of its preprocessing stages, our proposed tool can detect the synonyms, abbreviations or English computer terms. That's why; our tool can detect the similarities between texts easily.

4.9. Experiments with Student's Homeworks

In this experiment, student homeworks are compared with the proposed algorithm. The same homeworks are also compared with the existing plagiarism detection tools.

The random nine documents are selected from the second assignment of Database Organization and Management course. File1 represents the assignment of

the first user and File9 represents the assignment of the last user. The selected assignments are shown in Table 4.35. The assignments used in this experiment are compared with regard to similarity rate of all similarity tools mentioned in Section 4.7 in Table 4.36-4.55. The minimum match length is taken as the minimum value which is allowed by all similarity tools. For example, it is taken as 2 words in JPlag, WCopyfind, SIM and Sherlock, as 8 words in DOCCop, as 5 characters in CopyCatchGold, as 2 word-grams in Document Similarity, and as 3-grams in Ferret.

The similarity results of JPlag are shown as in Table 4.36 in the Appendices section. The minimum match length is taken as 2. JPlag calculates similarity taking into account the length of the files, so it calculates the similarity scores asymmetrically. So, the similarity rate of File1 to File2 is 78%, the similarity rate of File2 to File1 is 40%. Although JPlag has overlooked some similarity matches, these results are the quite close to realistic results.

The similarity results of Ferret are shown as in Table 4.37 in Appendices. The minimum match length is taken as 3, because Ferret does not allow changing the minimum match length and its default match length is 3. Ferret calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are both 33.02%. Ferret has overlooked many similarity matches, so these results are not the quite close to realistic results.

The similarity results of CopyCatch Gold are shown as in Table 4.38 in Appendices. The minimum match length is taken as 5, because the minimum match length choice of CopyCatch Gold is 5. But unlike JPlag and Ferret, the minimum match length type is not a word. Its type is character. CopyCatch Gold calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are both 63%. Copycatch Gold has overlooked a few similarity matches, so these results are considered closer to the realistic results.

The similarity results of WCopyfind are shown as in Table 4.39 in Appendices. The minimum match length is taken as 2. It allows changing the minimum match length to 1, but when it is selected as 1, and then the program produces incorrect results. For example, when the files are exactly the same, WCopyfind calculates the similarity rate greater than 100 percent. That's why; the

minimum match length is taken as 2. WCopyfind calculates similarity taking into account the length of the files, so it calculates the similarity scores asymmetrically like JPlag. For example, while the similarity rate of File1 to File2 is 77 percent, the similarity rate of File2 to File1 is 35 percent. Although WCopyfind has overlooked some similarity matches, these results are the quite close to realistic results.

The similarity results of SIM are shown as in Table 4.40 in Appendices. The minimum match length is taken as 2. SIM calculates similarity taking into account the length of the files, so it calculates the similarity scores asymmetrically like JPlag and WCopyfind. For example, while the similarity rate of File1 to File2 is 76 percent, the similarity rate of File2 to File1 is 36 percent. Although SIM has overlooked few similarity matches, these results are the quite close to realistic results.

The similarity results of Sherlock are shown as in Table 4.41 in Appendices. The minimum match length is taken as 2. Sherlock calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are both 33%. Sherlock has overlooked many similarity matches like Ferret, so these results are not close to realistic results.

The similarity results of Text_Matching Software of YTÜ are shown as in Table 4.42 in Appendices. The minimum match length is unknown, because this program does not allow changing of any setting. Text_Matching calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are both 71%. So these results are considered closer to the realistic results.

The similarity results of Cosine Similarity of YTÜ's Document Similarity Software are shown as in Table 4.43 in Appendices. The minimum match length is taken as 2. The Cosine Similarity of Document Similarity calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are both 57.34%. So these results are considered closer to the realistic results.

The similarity results of Jaccard Similarity of YTÜ's Document Similarity Software are shown as in Table 4.44 in Appendices. The minimum match length is taken as 2. The Jaccard Similarity of Document Similarity calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are

both 31.52%. The Jaccard Similarity of Document Similarity has overlooked many similarity matches, so these results are not close to the realistic results.

The similarity results of Dice Similarity of YTÜ's Document Similarity Software are shown as in Table 4.45 in Appendices. The minimum match length is taken as 2. The Dice Similarity of Document Similarity calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are both 47.94%. The Dice Similarity of Document Similarity has overlooked many similarity matches, so these results are not close to the realistic results.

The similarity results of Tanimoto Similarity of YTÜ's Document Similarity Software are shown as in Table 4.46 in Appendices. The minimum match length is taken as 2. The Tanimoto Similarity of Document Similarity calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are both 36.81%. The Tanimoto Similarity of Document Similarity has overlooked many similarity matches, so these results are not close to the realistic results.

The similarity results of our implemented Cosine Similarity are shown as in Table 4.47 in Appendices. There is no minimum match length setting. The Cosine Similarity calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are both 73%. So these results are considered closer to the realistic results.

The similarity results of our implemented Dice Similarity are shown as in Table 4.48 in Appendices. There is no minimum match length setting. It calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are both 63.34%. So these results are considered closer to the realistic results.

The similarity results of our implemented Jaccard Similarity are shown as in Table 4.49 in Appendices. There is no minimum match length setting. It calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are both 46.35%. The Jaccard Similarity has overlooked many similarity matches, so these results are not close to the realistic results.

The similarity results of DOCCop are shown as in Table 4.50 in Appendices. The minimum match length is taken as 8, because the minimum match length choice

of DOCCop is 8. DOCCop calculates the similarity scores symmetrically. So, the similarity rate of File1 to File2, and File2 to File1 are both 58%. DOCCop has overlooked many similarity matches, so these results are not the quite close to the realistic results.

The similarity results of Turnitin are shown as in Table 4.51 in Appendices. The minimum match length is unknown, because there is no minimum match length setting and this program does not allow changing of any setting. Turnitin calculates similarity taking into account the length of the files, so it calculates the similarity scores asymmetrically like JPlag, WCopyfind and SIM. So, the similarity rate of File1 to File2 is 0%, the similarity rate of File2 to File1 is 0%. Although Turnitin has overlooked some similarity matches, but among the found matches these results are not close to realistic results.

The similarity results of our proposed Greedy String Tiling Algorithm are shown as in Table 4.52 and Table 4.53 in Appendices. The minimum match length is taken as 2 in the results of the Table 4.52 and as 1 in the results of the Table 4.53. Our proposed Greedy String Tiling Algorithm calculates similarity taking into account the length of the files, so it calculates the similarity scores asymmetrically like JPlag, WCopyfind, SIM and Turnitin in which the results are the quite close to realistic results. So, as shown in the Table 4.52, the similarity rate of File1 to File2 is 88.13%, and the similarity rate of File2 to File1 is 44.71%. Also, as shown in the Table 4.53, the similarity rate of File1 to File2 is 77.7%, and the similarity rate of File2 to File1 is 39.42%. Because our proposed Greedy String Tiling Algorithm has considered many situations, it does not have overlooked similarity matches, so these results are considered quite closer to the realistic results, especially when the minimum match length was taken as 1.

The similarity results of our proposed Greedy String Tiling Algorithm for stopwords are shown as in Table 4.54 in Appendices. The minimum match length is taken as 2. Our proposed Greedy String Tiling Algorithm for stopwords calculates the similarity scores symmetrically like CopyCatch Gold and our implemented Cosine and Dice Similarity text measures. So, the similarity rate of File1 to File2, and File2 to File1 are both 99.21%. Our proposed Greedy String Tiling Algorithm

for stopwords does not have overlooked similarity matches, so these results are considered quite closer to the realistic results.

The similarity results of our proposed Greedy String Tiling Algorithm for punctuation marks are shown as in Table 4.55 in Appendices. The minimum match length is taken as 2. Our proposed Greedy String Tiling Algorithm for punctuation marks calculates the similarity scores symmetrically like CopyCatch Gold, our implemented Cosine and Dice Similarity text measures, and our proposed Greedy String Tiling Algorithm for stopwords. So, the similarity rate of File1 to File2, and File2 to File1 are both 94.12%. Our proposed Greedy String Tiling Algorithm for punctuation marks does not have overlooked similarity matches, so these results are considered quite closer to the realistic results.

To sum up, most algorithms mentioned above including Ferret, CopyCatchGold, Sherlock, Text_Matching Software of YTÜ, Document Similarity Software of YTÜ Kemik, our implemented Cosine, Dice, and Jaccard Similarities, and our proposed Greedy String Tiling Algorithm for stopwords and punctuation marks calculate similarity scores symmetrically, but the results of the tools running as asymmetrically including JPlag, WCopyfind, SIM, Turnitin, and our proposed Greedy String Tiling Algorithm as shown in the tables 4.36-4.55 are considered quite closer to the realistic similarity scores than the tools running as symmetrically. So, we improved our proposed Greedy String Tiling Algorithm both symmetrically and asymmetrically for text lines. Therefore, our proposed algorithm is considered quite successful.

The random ten matches are selected from the used assignments to show which algorithms are successful. The results are shown as in Table 4.56. F1 represents File1 and F9 represents the assignment of the last file File9. 1TO2 represents the similarity rate of the first file to the second file, and 2TO1 represents the similarity rate of the second file to the first file. Avg represents the average similarity scores of two files. According to these results, while Ferret and Sherlock produce unsuccessful results, JPlag, WCopyfind, SIM, Text_Matching Software of YTÜ, The Cosine Measure of YTÜ's Document Similarity Software, Turnitin, our implemented Cosine and Dice test similarity measures and our proposed GST

Algorithm produce accurate results. But among these successful tools, the results of our proposed GST Algorithm are the quite close to realistic results, especially when the minimum match length was taken as 1. The reason of its success, it uses many dictionaries and consists of important preprocessing phases.

Table 4.56. The Results of the Comparisons

	F1 – F7	**F2 – F3**	**F4 – F5**	**F4 – F6**	**F4 – F8**	**F5 – F1**	**F5 – F8**	**F6 – F1**	**F6 – F7**	**F8 – F9**
JPLAG1TO2	**96**	**99**	**99**	5	2	5	1	**96**	**96**	97
JPLAG2TO1	**78**	**98**	**81**	17	29	30	32	**95**	**77**	78
FERRET	71.7	94.5	81.2	0.18	0.07	0.9	0.1	89.3	70.4	85.1
COPYCATCH GOLD	**91**	**99**	**93**	23	16	23	16	**97**	**89**	**92**
WCOPYFIND1TO2	**94**	**98**	**99**	0	0	1	0	**95**	**95**	**100**
WCOPYFIND2TO1	**76**	**98**	**82**	4	6	10	8	**94**	**76**	**87**
SIM1TO2	**94**	**98**	**99**	4	1	**2**	1	**94**	**95**	96
SIM2TO1	**76**	**98**	**81**	17	17	**15**	22	**94**	**77**	85
SHERLOCK	69	94	78	0	0	0	0	91	66	79
TEXT_MATCHING	93	**99**	97	27	34	31	37	**99**	**90**	96
DOCUMENT SIMILARITY - COSINE	**88.9**	**98.7**	92.7	5.1	3.4	6.8	39	**96**	**87.3**	**92.4**
DOCUMENT SIMILARITY - JACCARD	73.7	96	81.8	0.8	0.6	1.4	0.7	88.8	74.2	84.7
DOCUMENT SIMILARITY - DICE	**84.9**	**98**	**90**	1.5	1.3	2.8	1.4	**94.1**	**85.2**	**91.7**
DOCUMENT SIMILARITY - TANIMOTO	78.3	**97.4**	85.3	1.9	0.8	2.5	0.8	92.4	75.9	85.4
COSINE	**93.7**	**99.5**	98.1	51.5	24.2	52.7	25.1	**97.9**	**92.3**	**94.1**
DICE	**92.7**	**99.5**	97.1	27.3	6.8	24.2	6.2	**97.9**	**91.5**	**93.8**
JACCARD	**86.5**	**99.0**	94.3	15.8	3.5	13.8	3.2	**96**	84.3	88.3
GSTM1 (Avg)	**87.6**	**99.3**	**89.1**	**19.1**	**10.7**	**17.7**	**9.4**	**96.6**	**86.1**	**91.1**
GSTM1_1TO2	**97.8**	**99.5**	**100**	**11.9**	**5.9**	**10.6**	**5.1**	**97.1**	**96.7**	**100**
GSTM1_2TO1	**79.3**	**99.1**	**80.3**	**49.5**	**65.7**	**54.3**	**70.6**	**96**	**77.6**	**83.6**
GSTM2 (Avg)	**86.3**	**98.9**	**89.1**	**4.5**	**3.7**	**5.3**	**3.1**	**95.5**	**85.1**	**91.1**
GSTM2_1TO2	**96.4**	**99.1**	**100**	**3**	**2.0**	**3.2**	**1.7**	**96**	**95.6**	**100**
GSTM2_2TO1	**78.1**	**98.7**	**80.3**	**12.4**	**22.6**	**16.2**	**23.5**	**95**	**76.7**	**83.6**
DOCCOP	**87**	**99**	91	0	0	3	0	**96**	**87**	**91**
TURNITIN1TO2	**96**	**99**	**100**	0	0	1	0	**96**	**96**	97
TURNITIN2TO1	**79**	**99**	81	0	0	5	0	**95**	**77**	**84**

4.10. The Comparison of the Running Time of All Similarity Tools

The running time of all tools are calculated and shown in the Table 4.57 in Appendices. Document Similarity Software of YTÜ and DocCop do not allow comparing all files at one time, that's why the running time could not be calculated. It compares two files in one second. However, users have to run these two tools for every pair of files. Therefore this table does not include these two programs. According to the Table 4.57, while the slowest softwares are our proposed GST Algorithm, and Turnitin, the fastest softwares are SIM, Sherlock and CopyCatch Gold, but the both successful and fastest softwares are GST Algorithm and SIM.

4.11. Using Data Mining for Determining the Effect of the User Behavior in Similarity Detection

The aims of this experiment were to discover relationships between the student's Moodle usage and the plagiarism behaviour obtained in the assignments of the course. In this experiment, the connection between student behavior in the logs of the Moodle e-learning system and the plagiarism case was analyzed by using classification methods. To do that Class Association Rules (CAR) and several classifiers from Weka data mining tool were used, and F-Measure values of the classifiers were compared.

The data used in this experiment was collected from the Data Organization and Management Moodle Course at Çukurova University Computer Engineering Department during the 2013-2014 semesters. The mdl_assignment, mdl_assignment_submission, mdl_course, mdl_course_modules, mdl_grade_grades, mdl_log, mdl_log_display, and mdl_user tables from Moodle were used. Collected logs in this thesis were preprocessed using the Knowledge Extraction based on Evolutionary Learning (Keel) program and classified using the Waikato Environment for Knowledge Analysis (Weka) program.

In the first stage, student logs were pre-processed. For instance, they were normalized with Min-Max Normalization and discretized with Uniform Frequency

using Keel. The data has 31 features, including identification number of student, identification number of assignment, new student info, number of login, number of assignment uploaded, assignment upload date before deadline, assignment view date after assignment add date, number of assignment viewed, number of total resource viewed, course enroll date, assignment grade, midterm scores, final scores, number of sentence used, number of word used, number of average word used, number of different word used, number of period used, number of comma used, number of line, number of semicolon used, number of question mark used, number of exclamation used, number of noun used, number of verb used, number of adjective used, number of pronoun used, number of preposition used, number of conjunction used, plagiarism class label, and class grade label.

In this experiment, all numeric attributes were discretized with equal frequency method and divided into three categories of low, medium and high. The 10-fold cross validation method method was used to split the dataset into train and test sets using Keel in the pre-processing phase.

After the pre-processing stage, in the second stage, the Apriori Algorithm with CAR was applied to each fold of the data for student behavior analysis. The metric type was confidence because of the using CAR. The minimum support was taken as 0.1 and a minimum confidence was taken as 0.7. The rule number was 100. The F-measure was calculated to measure the classification performance of the CAR rules to classify the test data. The weighted F-Measure values belong to 10 folds is given in Table 4.58.

Table 4.58. The comparison of Apriori tests

Data Set	**Classified Rate (%)**	**F-Measure**
The unnormalized & discretized dataset with all attributes	34.84	0.63
The unnormalized & discretized dataset with only assignment attributes	33.55	0.63
The unnormalized & discretized dataset with only web attributes	58.06	0.55
The normalized & discretized dataset with all attributes	43.23	0.63
The normalized & discretized dataset with only assignment attributes	37.42	0.62
The normalized & discretized dataset with only web attributes	58.06	0.56

As shown in Table 4.58, the classified test instances rate is high when we used the normalized/unnormalized and discretized data sets with only web attributes. When the classified rate is low, the F-Measure is high. According to this table, the best results are revealed when the normalized dataset with only web attributes is used. In that case, the classified rate of this dataset is 58% and the weighted F-Measure of classification for the classified test instances is 56%.

According to this confidence value given above, some of the important rules and their confidences are shown in Table 4.59:

Table 4.59. The most important Apriori Rules for the datasets

Total_Resource_View_Number=high 4 ==> Plagiarism=no 4 conf:(1.0)
New_Student=yes Total_Resource_View_Number=high 4 ==> Plagiarism=no 4 conf:(1.0)
Total_Resource_View_Number=medium Assignment_Grade=high 4 ==> Plagiarism=yes 4 conf:(1.0)
Assignment_Grade=medium 3 ==> Plagiarism=no 3 conf:(1.0)
Course_Enrol=high Different_Word=low Question_Mark= low Exclamation= low 17 ==> Plagiarism=yes 15 conf:(0.88)
Assignment_View_Date_After_Assignment_Add_Date= high Course_Enrol= low 16 ==> Plagiarism=no 14 conf:(0.88)
Assignment_Grade=medium 3 ==> Plagiarism=no 3 conf:(1)
Assignment_ID=o158 Assignment_View_Date_After_Assignment_Add_Date=medium 3 ==> Plagiarism=yes 3 conf:(1)
Assignment_ID=o158 Total_Resource_View_Number=medium 3 ==> Plagiarism=yes 3 conf:(1)

These rules show that, generally there are more meaningful rules for the class “Plagiarism=no”. The rules of the class “Plagiarism=yes” are not very meaningful, so some experiments are done with the other classifiers.

In the other experiment, some tests are performed to determine the best feature set and classifier for plagiarism detection. The data was used in 3 versions: all attributes, only web attributes and only assignment attributes. Attribute values are used either normalized or unnormalized, and all applicable classifiers are used in Weka tool. Then the best classifier is determined in terms of F-Measure value.

In this experiment, 10-fold cross validation is applied to the data set in Weka. All applicable classification algorithms are as shown in Table 4.60. The F-measure of best classifiers for the raw and normalized attribute values of the dataset with all, only web, and only assignment attributes are shown as in Table 4.61.

According to Table 4.61, KStar gives the highest F-Measure when we use 10-fold cross validation on the raw dataset with only assignment attributes in that case, we can predict plagiarism with 70% accuracy without using any plagiarism detection tool. The best classification accuracy is obtained when only attributes belong to assignments are used. If we only use web usage attributes, accuracy reduces to 62%. This result occurs because our students do not use Moodle activities effectively.

Table 4.61. The comparison of all experiments for plagiarism with 10-fold cross validation

Dataset	**Algorithm**	**F-Measure (10-fold cross validation)**
Raw, all, numeric	ADTree	0.632
Raw, assignment, numeric	KStar	0.702*
Raw, web, numeric	LADTree, ADTree	0.603
Normalized, all, numeric	ADTree	0.652
Normalized, assignment, numeric	KStar	0.695
Normalized, web, numeric	JRip	0.625
Raw, all, categoric	DecisionStump	0.622
Raw, assignment, categoric	IBK (k=3)	0.613
Raw, web, categoric	DecisionStump	0.622
Normalized, all, categoric	DecisionStump	0.622
Normalized, assignment, categoric	KStar	0.618
Normalized, web, categoric	DecisionStump	0.622

*successful

4.12. Using Data Mining for Determining the Effect of the User Behavior in Grade Prediction

The aims of this experiment were to discover relationships between the student's web usage and the grade obtained in the course. In this experiment, the connection between student behavior in the logs of the Moodle e-learning system and the passing grade was analyzed using Apriori Algorithm. Class association rules (CAR) was used instead of the general association rules in this experiment. The dataset, preprocessing methods, and classifiers in Section 4.11 were used in this experiment. The 10-fold cross validation method method was used to split the dataset into train and test sets.

The minimum support was taken as 0.1 and a minimum confidence was taken as 0.7. The rule number was 100. The F-measure was calculated to measure the

performance of the test data rules. The comparison of test data rules is shown as in Table 4.62.

Table 4.62. The comparison of Apriori tests

Data Set	Classified Rate (%)	F-Measure
The raw dataset with all attributes	100	0.96
The raw dataset with only assignment attributes	100	0.96
The raw dataset with only web attributes	92.90	0.98
The normalized dataset with all attributes	100	0.96
The normalized dataset with only assignment attributes	100	0.96
The normalized dataset with only web attributes	92.26	0.98

As shown in Table 4.62, the classified rate is high when we used either the raw data sets or the normalized data sets with all attributes or only assignment attributes. Both the classified rate and the F-Measure of classification are high. So, in all cases F-Measure is greater than or equal to 96%. According to this table, the optimum results are revealed when the normalized/unnormalized dataset with all attributes or only assignment attributes are used.

According to the results, the F-Measure value of the classification by using CAR had a high value. This means that these rules can help to classify successful students depending on the activities carried out in Moodle and identify unsuccesful students depending on the user behavior and activities completed in Moodle in the rate of 96% at the beginning of the semester. Therefore, the web usage mining methods can be used to discover useful knowledge from Moodle data of students before the end of the semester.

According to this confidence value given above, some of the important rules and their confidences are shown in Table 4.63.

Table 4.63. The most important Apriori Rules for the datasets

Assignment_Upload_number=medium 86 ==> Grade=pass 86 conf:(1.0)
Assignment_Upload_number=medium Assignment_Grade=high 50 ==> Grade=pass 50 conf:(1.0)
Final= medium 51 ==> Grade=pass 51 conf:(1.0)
Login_Number=high 53 ==> Grade=pass 53 conf:(1.0)
Midterm= medium 44 ==> Grade=pass 44 conf:(1.0)
Midterm=high 47 ==> Grade=pass 47 conf:(1.0)

In the other experiment, some tests are performed to determine the best feature set and classifier for plagiarism detection. The data are used in 3 versions: all attributes, only web attributes and only assignment attributes. Attribute values are used either normalized or unnormalized, and all applicable classifiers are used in Weka tool. Then the best classifier is determined in terms of F-Measure value.

In this experiment, 10-fold cross validation is applied to the data set in Weka. The comparison of F-measure of all classification algorithms for raw and normalized attribute values of the dataset with all, only web, and only assignment attributes are shown as in Table 4.64.

According to Table 4.64, we can obtain the highest F-Measure when we use the all attributes in their numeric form as raw and/or normalized, when we use the only web attributes in their numeric and/or categoric form as raw and/or normalized. So the web attributes usage gives the highest F-Measure when 10-fold cross validation is done. According to these results, the most successful algorithms are NNge, a rule based classification algorithm, and LADTree, a tree based classification algorithm. Also, BayesNet, a Bayesian classification algorithm, and SMO, a function based classification algorithm are considered successful algorithms. While the prediction of plagiarism behavior is not successful – with 70% accuracy – the prediction of passing grade is very successful – with 99% accuracy. So, it can be predicted the failure students before the ending of the semester and can be given feedbacks to these students.

Table 4.64. The comparison of all experiments in web usage mining for grade

Dataset	**Algorithm**	**F-Measure (10-fold cross validation)**
Raw, all, numeric	BayesNet, NNge, and SMO	0.994*
Raw, assignment, numeric	FT, JRip, LADTree, LMT, MultilayerPerceptron, NNge, Ridor, SimpleLogistic, SMO, and VFI	0.987
Raw, web, numeric	BayesNet and NNge	0.994*
Normalized, all, numeric	MultilayerPerceptron, NNge, and SMO	0.994*
Normalized, assignment, numeric	FT, LADTree, LMT, MultilayerPerceptron, NNge, Ridor, SimpleLogistic, SMO, and VFI	0.987
Normalized, web, numeric	NNge	0.994*
Raw, all, categoric	FT	0.987
Raw, assignment, categoric	VFI	0.98
Raw, web, categoric	LADTree	0.994*
Normalized, all, categoric	FT	0.981
Normalized, assignment, categoric	VFI	0.98
Normalized, web, categoric	LADTree	0.994*

*successful

4.13. The Comparison of the Proposal with the Related Studies

In the literature, there exist several studies which include plagiarism detection of programming assignments and textual documents into distance education systems. In this section, we give a brief review of these studies and we describe the differences of our study from the existing systems in the last part of this section. A brief summary and comparison of the related studies about the plagiarism detection and our proposed system is also presented in Table 4.65.

According to the comparison results, LMS systems have been used in many studies. In some studies, plagiarism detection tools or plagiarism detection algorithms have been proposed. These tools or algorithms have been used for source code or textual documents. Most of the algorithms used for textual document similarity detection are distance based algorithms and token based algorithms. Most of them have been used for English language. We have included plagiarism detection tool into our Moodle based distance education system to decrease the time and effort needed for grading of the textual assignments submitted by our students written in Turkish language. Our algorithms are based on both distance based algorithms and token based algorithms.

We also give a brief review of the related studies about the web usage mining and we describe the differences of our study. The comparison of our proposal with the related studies about the web usage mining is shown in Table 4.66.

According to the comparison results, our proposal has originality, because none of the previous studies have analyzed the submitting similar assignments behavior of the students by some web usage mining methods, they have only aimed to analyze and group the learning behavior of students by final grade. They have not take into consideration the plagiarism information of the students but the plagiarism behaviour of the students was very important for our thesis. We have aimed to analyze the relationship between the student's behavior and the submitting similar assignments result. We also aim to analyze the relationship between the student's behavior and the final grade like the other studies. Our thesis analyzed the submitting similar assignments behavior of the students by some web usage mining methods and that's why in that respect our thesis is an original study.

5. CONCLUSIONS

In this thesis, it is observed that there is a need for a system which can detect student's plagiarism. So, some dictionaries for preprocessing phase are developed, and a string matching algorithm, Greedy String Tiling, a distance metric, Cosine Similarity are used to detect plagiarism in Turkish documents of student assignments in Moodle, a distance education environment, and data mining methods are used. Such a free plagiarism detection system for Turkish documents is not currently available for Moodle. Our main contributions to plagiarism detection systems are that our system consists in detecting synonym, abbreviation and glossary of computer terms replacements. Also the other contribution is that our system consists of a similarity metric, Cosine Similarity, as a threshold to pass up unnecessary comparisons and to improve the running time. Thus, an approximate string matching algorithm and a similarity metric were used together to detect similarities of the Turkish documents.

In our system, after the preprocessing phases — which includes punctuation removal, numbers and whitespaces removal, changing to lowercases of characters, using abbreviation list and glossary of computer terms — the Cosine Similarity produces a similarity score. If the score is higher than the threshold score, the second preprocessing phase — which includes stemming, using a synonym dictionary, and stopword removal — is done and the proposed Greedy String Tiling Algorithm is applied to the textual documents to produce a similarity match score. Then the proposed Greedy String Tiling Algorithm is applied to punctuations and stopwords. The similarity score of the Greedy String Tiling experimental evaluation shows that the proposed algorithm is effective on detecting similarities for the Turkish language, especially for computer science related documents written in Turkish. This proposed method can be easily used to detect plagiarism on the textual assignments that are written in Turkish, and this algorithm can be easily included into Moodle. This helps to increase the quality of computer engineering education by forcing students to create their own assignments.

We observe that, in respect to processing time, our algorithm is fast like the other algorithms, but gives more successful and accurate results on short assignments, which have roughly between 1 and 350 lines. In long assignments which have more than 350 lines, our algorithm gives successful results but the process time is slower than the other algorithms analyzed in Results and Discussion section.

As a future work, the resulting publications of these studies will contribute to the use of plagiarism detection tools in Turkish. It is aimed at contributing to become conscious about a new country-wide progress for finding solutions for plagiarism detection problems which are performed much more around the world. Also, we plan to study how to decrease the processing time in more detail in order to improve our algorithm to use effectively. The proposed algorithms written in this study are strong with the preprocessing stages because of the existence of several dictionaries including synonyms dictionaries, abbreviation list, phrase list, and glossary of computer terms. These dictionaries can be made more specific or more comprehensive in order for the proposed algorithm to be further strengthened. In the future, the developed system, in this thesis can be improved to be used for other languages. This is why we plan to add the language choice to our proposed algorithm.

REFERENCES

AGNES, M., E., 1999. Webster's New World College Dictionary, ISBN: 978-0028631189, Fourth Edition.

AGRAWAL, R., SRIKANT, R., 1994. Fast algorithms for mining association rules, In Proc. 20th Int. Conf. Very Large Data Bases, VLDB, Morgan Kaufmann, 487–499.

AGRAWAL, R.., IMIELINSKI T., SWAMI A. N., 1993. Mining association rules between sets of items in large databases, In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 207-216.

AJOUDANIAN, S. and JAZI, M. D., 2009. Deep Web Content Mining, World Academy of Science, Engineering and Technology, 49.

AKIN, M. D., AKIN, A. A., 2007. Türk Dilleri için Açık Kaynaklı Doğal Dil İşleme Kütüphanesi: ZEMBEREK, Elektrik Mühendisliği, 431, 38-44.

AKTAŞ, Ö., 2006. Türkçe için Verimli bir Cümle Sonu Belirleme Yöntemi, Akademik Bilişim 2006 - Bilgi Teknolojileri Kongresi IV, Pamukkale University, Denizli, Türkiye.

AKTAŞ, Ö., BİRANT, Ç. C., AKSU, B., ÇEBİ, Y., 2013. Automated Synonym Dictionary Generation Tool for Turkish (ASDICT), BILIG - Türk Dünyası Sosyal Bilimler Dergisi, NUMBER 65, 47-68.

AL, U., MADRAN, R. O., 2004. Web Tabanlı Uzaktan Eğitim Sistemleri: Sahip Olması Gereken Özellikler ve Standartlar, Bilgi Dünyası, 5(2): 259-271.

ALHASHIM, A. F., 2005. Software Development of Data Compression Algorithm (Application to Turkish Text), Master Thesis, Near East University Department of Computer Engineering.

ALI, A. M. E. T., ABDULLA, H. M. D., SNÁSEL, V., 2011. Overview and Comparison of Plagiarism Detection Tools, In Proceedings of DATESO, pp: 161-172.

ALI, A. M. E. T., ABDULLA, H. M. D., SNÁSEL, V., 2011. Survey of Plagiarism Detection Methods. In Proceedings of the 5th Asia Modelling Symposium (AMS), Manila, Philippines, pp. 39–42.

AMASYALI, M. F., 2005. Türkçe WordNet'in Otomatik Oluşturulması, SIU 2005, Kayseri.

AMASYALI, M. F., BEKEN, A., 2009. Türkçe Kelimelerin Anlamsal Benzerliklerinin Ölçülmesi ve Metin Sınıflandırmada Kullanılması, SIU 2009, Antalya, Türkiye.

AMEELA, T., KALEESWARAN, D., 2013. Credential Clustering in Parallel Comparability Frequency Amplitude, International Journal of Advanced Computer Research, Volume: 3, Number: 1, pp. 50-55.

ANTWORTH, E., 1990. PC-KIMMO: A Two-level Processor for Morphological Analysis, Summer Institute of Linguistics, Inc.

ARISOY, E., SAK, H., SARAÇLAR, M., 2007. Language Modeling for Automatic Turkish Broadcast News Transcription, In Proc. Interspeech-Eurospeech, Antwerp, Belgium, pp: 2381–2384.

ARWIN, C., TAHAGHOGHI, S. M. M., 2006. Plagiarism detection across programming languages. Proceedings of the 29th Australasian Computer Science Conference - Volume 48 (Darlinghurst, Australia, 2006), 277–286.

ATTRIBUTOR, www.attributor.com, Retrieved: 26.01.2013.

BALKANET, http://www.dblab.upatras.gr/balkanet/, Retrieved: 26.02.2014.

BARNARD, S., BRORSON, B., IVANOVA, E., 2007. Moodle: Why? Where? When? How?, University of Minnesota.

BEJARANO, A. M., GARCÍA, L. E., ZUREK, E. E., 2013. Detection of source code similitude in academic environments. Computer Applications in Engineering Education, DOI: 10.1002/cae.21571.

BELAL, A., OWAIDAT, B., SALEH, N., JABER, R., 2012. Error correction capabilities in block ciphers, 2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), pp.66,69, DOI: 10.1109/ICTEA.2012.6462905.

BERGHEL, H. L. and SALLACH, D. L., 1984. Measurements of program similarity in identical task environments. ACM SIGPLAN Notices, 19(8):65–76

BEYAZPERDE, 2014. http://www.beyazperde.com/

BHOSALE, S. P., 2014. Novel Approach to Discover High Utility Itemsets from Transactional Database, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3, Issue 1, pp. 139-142.

BİLGİN, O., ÇETİNOĞLU, Ö., OFLAZER, K., 2004a. Building a Wordnet for Turkish, Romanian Journal of Information Science And Technology Volume 7, Numbers 1–2, 163–172.

BİLGİN, O., ÇETİNOĞLU, Ö., OFLAZER, K., 2004b. Morphosemantic Relations In and Across Wordnets: A Study Based on Turkish, in Proceedings of the Global WordNet Conference, Masaryk University, Brno, Czech Republic.

BİRANT, Ç. C., 2008. Root-Suffix Seperation of Turkish Words, Yüksek Lisans Tezi, Dokuz Eylül Üniversitesi Bilgisayar Mühendisliği, İzmir.

BLACK, P. E., 2006. Manhattan distance, in Dictionary of Algorithms and Data Structures [online], Vreda Pieterse and Paul E. Black, eds., Available from: http://www.nist.gov/dads/HTML/manhattanDistance.html

BOSS SUBMISSION SYSTEM, 2014. http://www.dcs.warwick.ac.uk/boss/history.php

BULL, J., 2001. Technical review of plagiarism detection software report, CAA, University of Luton

BURROWS, S., TAHAGHOGHI, S. M. M., ZOBEL, J., 2006. Efficient Plagiarism Detection for Large Code Repositories. School of Computer Science and Information Technology, RMIT University, Melbourne, Australia, pp 158-159

BUTAKOV, S., & SHCHERBININ, V., 2009. On the number of search queries required for Internet plagiarism detection. In Proceedings of the 2009 Ninth IEEE International Conference on Advanced Learning Technologies, 482–483.

BÜYÜK TÜRKÇE SÖZLÜK, http://www.tdk.gov.tr/index.php?option=com_bts, Retrieved: 25.01.2015.

CAMPBELL, S., 2006. Perceptions of mobile phones in college classrooms: ringing, cheating, and classroom policies. Communication Education, 55(3), 280-294.

CAN, F., KOÇBERBER, S., BALÇIK, E., KAYNAK, C., ÖCALAN, H. Ç., VURSAVAŞ, O. M., 2008. Information retrieval on Turkish texts. 407-421.

CANTONE, D., FARO, S., GIAQUINTA, E., 2012. Adapting Boyer-Moore-Like Algorithms for Searching Huffman Encoded Texts, Int. J. Found. Comput. Sci., pp.343-356.

CARUS, A., ERSİN, A. K., MESUT, A., 2007. Dizgi Eşleme Algoritmalarının Alfabeye Bağlı Etkinliklerinin Araştırılması, Elektrik-Elektronik-Bilgisayar Mühendisliği 12. Ulusal Kongresi, Eskişehir.

CHAN, A. T. S., CAO, J., LIU, C. K., CAO, W., 2004. VPL: An Online Distance Learning Platform for Virtual Programming Laboratory, International Journal of Computer Processing of Languages, 17:01, 41-59

CHANG, W., MARR, T., 1994. Approximate string matching and local similarity, In Proc. 5th Combinatorial Pattern Matching (CPM'94), Lecture Notes in Computer Science (LNCS) 807, pp. 259–273.

CHARRAS, C., LECROQ, T., 1997. Exact String Matching Algorithms Animation in Java,ESMAJ: http://www-igm.univ-mlv.fr/~lecroq/string/

CHARRAS, C., LECROQ, T., 2004. Handbook of Exact String Matching Algorithms, College Publications, ISBN: 978-0-9543006-4-7.

CHEN, C. Y., YEH, J. Y., KE H, R., 2010. Plagiarism Detection using ROUGE and WordNet, Journal of Computing 2:34-44.

CHEN, M., HAN, J., YU, P., 1996. Data Mining: An Overview From A Database Perspective, IEEE Transactions on Knowledge and Data Engineering, Vol.8, No. 6, pp. 866-883.

CHIEF, L. C., 2000. What can near synonyms tell us?, Computational linguistics and Chinese Language Processing (5/1): 47-60.

CHIMPSKY, http://chimpsky.uwaterloo.ca/, Retrieved: 26.01.2013.

ÇİLDEN, E. K., 2006. Stemming Turkish Words Using Snowball, http://snowball.tartarus.org/algorithms/turkish/stemmer.html, Retrieved: 31.03.2014.

CLOUGH, P., 2003. Old and new challenges in automatic plagiarism detection. National UK Plagiarism Advisory Service (Online): http://ir.shef.ac.uk/cloughie/papers/pas_plagiarism.pdf

CODEMATCH, 2014. http://www.safe-corp.biz/products_codematch.htm

COLE, J., FOSTER, H., 2008. Using Moodle: Teaching with the Popular Open Source Course Management System, 2nd Edition, O'Reilly Community Press, p. ix

ÇÖLTEKİN, Ç., 2010. A freely available morphological analyzer for Turkish, Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), Valletta, Malta, European Language Resources Association (ELRA).

COMAS, R., & SUREDA, J. 2008. El ciberplagi acadèmic: esbrinant-ne les causes per tal d'enllestir les solucions. Digithum(10), 1-6.

COMAS-FORGAS, R., & SUREDA-NEGRE, J. 2010. Academic Plagiarism: Explanatory Factors From Students' Perspectives. Journal of Academic Ethics/Springer Science and Business Media, 8(3), 217-232.

COMPILATIO, Access Date: 2013.

COOLEY, R., MOBASHER, B., SRIVASTAVA, J., 1997. Web Mining: Information and Pattern Discovery on the World Wide Web. In Proc. of the 9th IEEE Intl. Conf. on Tools With Artificial Intelligence (ICTAI'97), pp. 558-567.

COPYCATCH GOLD, CFL Software Limited, 2012, http://www.cflsoftware.com/GoldFull.html, Retrieved: 20.09.2014.

COPYSCAPE, http://www.copyscape.com/, Retrieved: 26.01.2013.

COSMA, G., JOY, M., 2006. Source-code Plagiarism: A UK Academic Perspective. Research Report No. 422. Department of Computer Science, University of Warwick. http://www.dcs.warwick.ac.uk/reports/422.html

COSMA, G., JOY, M., 2012. An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis, Computers, IEEE Transactions on , vol.61, no.3, pp.379,394, DOI: 10.1109/TC.2011.223.

CROT PRO, Access Date: 2013. https://moodle.org/plugins/view.php?plugin=plagiarism_crotpro

CROT, Access Date: 2013. https://moodle.org/plugins/view.php?plugin=plagiarism_crot

ÇÜ_CEN_Moodle, 2009. Çukurova University's Department of Computer Engineering Moodle web site, http://bmb.cu.edu.tr/moodle/bmb/moodle/

CULWIN, F., MACLEOD, A., LANCASTER, T., 2001. Source code Plagiarism in UK HE Computing Schools, Issues, Attitudes and Tools, Technical Report No. SBU-CISM-01-02, South Bank University

DELVIN, M., 2002. Plagiarism detection software: how effective is it?, Assessing Learning in Australian Universities, http://www.cshe.unimelb.edu.au/assessinglearning/docs/PlagSoftware.pdf

DEZA, E., DEZA, M. M., 2009. Encyclopedia of Distances, page 94, Springer.

DICE, L. R., 1945. Measures of the Amount of Ecologic Association Between Species, Ecology 26 (3): 297–302. doi:10.2307/1932409, JSTOR 1932409.

DİL LİSTESİ: TÜRKÇE SÖZLÜK (2005)'TEKİ SÖZLERİN KÖKENLERİNE AİT SAYISAL DÖKÜM, TDK, http://web.archive.org/web/20070301064559/http://www.tdk.gov.tr/TR/BelgeGoster.aspx?F6E10F8892433CFFAAF6AA849816B2EF1A46C5FBFA979D0C, Retrieved: 25.01.2015.

DONNELY, C., 1994. Linguistics for Writers, Albany, State University of New York Press.

DUNHAM, M. H., 2003. Data Mining Introductory and Advanced Topics, Pearson Education.

DURSUN, B., 2008. Türkçe Metinlerin Benzerliğinin Hesaplanması İçin Yeni Bir Yöntem, Signal Processing, Communication and Applications Conference, SIU 2008, IEEE 16th, p. 1-4.

DZIUDA, D. M., 2010. Data Mining for Genomics and Proteomics: Analysis of Gene and Protein Expression Data, John Wiley & Sons, 328 pages.

EKMEKÇIOĞLU, F. Ç., LYNCH, M. F., WILLETT, P., 1996. Stemming and N-gram matching for term conflation in Turkish texts, Information Research, Vol. 2 No. 2.

EL TAHİR ALİ, A. M., DAHWA ABDULLA, H. M., & SNÁSEL, V. 2011. Overview and Comparison of Plagiarism Detection Tools. In Proceedings of DATESO, 161-172.

EL-HALEES, A., 2009. Mining students data to analyze e-Learning behavior: A Case Study, The 2008 International Arab Conference of Information Technology (ACIT2008) – Conference Proceedings, University of Sfax, Tunisia, Dec 15- 18.

EL-KAHLOUT, İ. D., OFLAZER, K., 2004. Use of WordNet for Retrieving Words from their Meanings, in Proceedings of the Global WordNet Conference, Masaryk University, Brno, Czech Republic.

EL-KAHLOUT, İ. D., OFLAZER, K., 2006. Türkçe-İngilizce için istatistiksel bilgisayarlı çeviri sistemi. In: TAINN 2006 Fifteenth Turkish Symposium on Artificial Intelligence and Neural Networks, Akyaka, Muğla, Türkiye.

ERSİN, A. K., 2008. Dizgi Eşleme Algoritmalarının İncelenmesi ve Yeni Bir Dizgi Eşleme Algoritması, Yüksek Lisans Tezi, Trakya Üniversitesi Bilgisayar Mühendisliği, Edirne.

ETBLAST: A TEXT-SİMİLARİTY BASED SEARCH ENGİNE, http://etest.vbi.vt.edu/etblast3/, Retrieved: 26.01.2013.

EVE PLAGIARISM DETECTION SYSTEM, http://www.canexus.com/, Retrieved: 26.01.2013.

FAIDHI, J. A. W. and ROBINSON, S. K., 1987. An empirical approach for detecting program similarity within a university programming environment. Computers and Education, 11(1):11–19.

FARO, S., & LECROQ, T. 2013. The Exact Online String Matching Problem: A Review of the Most Recent Results. ACM Computing Surveys, 45(2), 13:1-13:42.

FOXLEY, E., 1999. Ceilidh Documentation on the World Wide Web, Retrieved 15.11.2004, from http://www.cs.nott.ac.uk/*ceilidh/papers.html.

FREDRIKSSON, K., & NAVARRO, G. 2004. Average-Optimal Single and Multiple Approximate String Matching. ACM Journal of Experimental Algorithmics (JEA), 9 (Article 1.4), 1-47.

GABRIELA, G., 2013, Opportunities To Improve The Management Methods Of Romanian Organizations, The Annal of The University of Oradea Economic Sciences, No:1, pp. 1682-1689.

GARCÍA, E., ROMERO, C., VENTURA, S., CASTRO, C., CALDERS, T., 2010. Association Rule Mining in Learning Management Systems, Handbook of Educational Data Mining, CRC Press, Pages 93–106.

GARG, M., & SINGH, V. 2014. Plagiarism and Anti-Plagiarism Software. EUROPEAN ACADEMIC RESEARCH, 1(10), 3266-3273.

GETOOR L., 2003. Link Mining: A New Data Mining Challenge, SIGKDD Explorations, Volume: 4, Issue:2.

GIANNAKOURIS – SALALIDIS, V., PLEROU, A., SIOUTAS, S., 2014. CSMR: A Scalable Algorithm for Text Clustering with Cosine Similarity and MapReduce, Artificial Intelligence Applications and Innovations, 211-220.

GIPP, B., MEUSCHKE, N., 2011. Citation Pattern Matching Algorithms for Citation-based Plagiarism Detection: Greedy Citation Tiling, Citation Chunking and Longest Common Citation Sequence, Proceedings of the 11th ACM Symposium on Document Engineering (DocEng2011), ACM, pp. 249–258, doi:10.1145/2034691.2034741, ISBN 978-1-4503-0863-2

GITCHELL, D., TRAN, N., 1999. Sim: a utility for detecting similarity in computer programs. In The proceedings of the thirtieth SIGCSE technical symposium on Computer science education (SIGCSE '99). ACM, New York, NY, USA, 266-270. DOI=10.1145/299649.299783 http://doi.acm.org/10.1145/299649.299783

GLATT PLAGIARISM SCREENING PROGRAM AND SERVICE, http://www.plagiarism.com, Retrieved: 26.01.2013.

GOMES, L., 2006. Some Students Use Net To Hire Experts to Do Their School Work. The Wall Street Journal [Online], http://online.wsj.com/public/us

GORDON, R. G., 2005. Ethnologue: Languages of the World, Fifteenth edition. Language Family Trees – Altaic, Retrieved: 18.03.2007.

GOTOH, O., 1982. An improved algorithm for matching biological sequences. Journal of molecular biology 162: 705.

GRAMMARLY, http://www.grammarly.com/, Retrieved: 26.01.2013.

GRUNE, D., HUNTJENS, M., 1989. Het detecteren van kopieën bij informatica-practica (English Translation: Detecting copied submissions in computer science workshops), Informatie, 31, 11, pp. 864-867

GRUNER, S. and NAVEN, S., 2005. Tool Support for Plagiarism Detection in Text Documents, Proceedings of the ACM Symposium on Applied Computing (SAC'05), March 13-17, Santa Fe, New Mexico, USA.

GÜNCEL TÜRKÇE SÖZLÜK, TDK, http://web.archive.org/web/20070321023726/http://www.tdk.gov.tr/TR/BelgeGoster.aspx?F6E10F8892433CFFAAF6AA849816B2EFB40CE59E171C629F, Retrieved: 25.01.2015.

GÜNDÜZ M., BAYKAN, Ö.K., YILDIZ, F., 2007. Elektronik Deneyleri İçin Sanal Laboratuar Uygulaması, Journal of Technical-Online, Volume 6, Number: 2.

GÜNGÖR, T., KURU, S., 1993. Representation of Turkish Morphology in ATN, Proceedings of Second Symposium on Artificial Intelligence and Artificial Neural Networks, pp. 92-104, İstanbul.

GÜNGÖRDÜ, Z., OFLAZER, K., 1995. Parsing Turkish using the lexical functional grammar formalism, Machine Translation, vol. 10, pp. 293-319.

GUPTA, D., BHATIA, K. K., SHARMA, A. K., 2009. A Novel Indexing Technique for Web Documents using Hierarchical Clustering, IJCSNS International Journal of Computer Science and Network Security, VOL.9, No.9, pp. 168-175.

GUPTA, G. K., 2014. Introduction to Data Mining with Case Studies, New Delhi : PHI Learning, 3rd Ed., ISBN-ISSN: 9788120350021.

GUTIÉRREZ, E., TRENAS, M.A., RAMOS, J., CORBERA, F., ROMERO, S., 2009. A new Moodle module supporting automatic verification of VHDL-based assignments. Computers and Education, 54(2):562–577.

HACKER, D., 2007. A Writer's Reference, 6th Edition, Boston: Bedford/St. Martin's.

HAGE, J., RADAMEKER, P., VUGHT, N. V., 2010. A comparison of plagiarism detection tools, Technical Report UU-CS-2010-015, ISSN: 0924-3275.

HAKKANİ, D. Z., OFLAZER, K., ÇİÇEKLİ, İ., 1996. Tactical Generation in a Free Constituent Order Language, Proceedings of 1996 International Workshop on Natural Language Generation, Herstmonceux, Sussex, UK.

HAMMING, R. W., 1950. Error detecting and error correcting codes, Bell System Technical Journal 29 (2): 147–160, doi:10.1002/j.1538-7305.1950.tb00463.x, MR 0035935.

HAN, J., KAMBER, M., 2006. Data Mining Concepts and Techniques, 2nd ed., Morgan Kaufmann Publishers, San Francisco, 800 pages.

HARIS, Z. S., 1968. Mathematical structures of language, Wiley, pp. 12.

HD FİLM VADİSİ, 2014. http://www.hdfilmvadisi.com/

HECKEL, P., 1978. A Technique For Isolating Differences Between Files, Communication of the ACM, Volume: 21, Number: 4.

HERRANZ, J., NIN, J., 2009. Partial Symbol Ordering Distance, Modeling Decisions for Artificial Intelligence, Lecture Notes in Computer Science, Springer Berlin Heidelberg, Volume 5861, pp. 293-302.

HEXHAM, I., 1999. The Plague of Plagiarism. http://www.ucalgary.ca/~nurelweb/academic/plag.html
https://moodle.org/plugins/view.php?plugin=plagiarism_compilatio

HUANG, C. R. and HONG, J. F., 2005. Deriving Conceptual Structures Form Sense: A Study of Near Synonymous Sensation Verbs, Journal of Chinese Language and Computing (JCLC). Volume 15. No 3. Singapore.

INAMDAR, S. A. and SHINDE, G. N., 2010. An Agent Based Intelligent Search Engine System for Web Mining, International Journal on Computer Science and Engineering, Vol. 02, No. 03.

IPARADIGMS, 2006. Plagiarism Prevention: Stop plagiarism now… with Turnitin®, Product datasheet, http://turnitin.com/static/pdf/datasheet_plagiarism.pdf

IŞIK M., ÇAMURCU, A. Y., 2008. Web Belgeleri Kümelemede Benzerlik ve Uzaklık Ölçütleri Başarılarının Karşılaştırılması, Marmara Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 20, 35-49.

ITHENTICATE, http://www.ithenticate.com, Retrieved: 26.01.2013.

JACOBS N., 2004. Relational Sequence Learning and User Modelling, Katholieke Universiteit Leuven, Faculteit Wetenschappen, Faculteit Toegepaste Wetenschappen; Departement Computerwetenschappen, Leuven, xvi + 235 pp.

JARO M. A., 1989. Advances in record linkage methodology as applied to the 1985 census of Tampa Florida, Journal of the American Statistical Society 84 (406): 414–20.

JARO M. A., 1995. Probabilistic linkage of large public health data file, Statistics in Medicine 14 (5-7): 491–8, doi:10.1002/sim.4780140510, PMID 7792443.

JIFFRIYA, M., AKMAL JAHAN, M., RAGEL, R. R., & DEEGALLA, S. 2013. AntiPlag: Plagiarism detection on electronic submissions of text based assignments. Industrial and Information Systems (ICIIS), 8th IEEE International Conference on, pp. 376 - 380.

JOKINEN, P., TARHIO, J., UKKONEN, E., 1996. A comparison of approximate string matching algorithms, Software - Practice and Experience 26, 12, 1439-1458.

KAFKA, F., 1915. Dönüşüm (Die Verwandlung).

KALAIVENDHAN, K., SUMATHI, P., 2014. An Efficient Clustering Method To Find Similarity Between The Documents 2, 2532–2535.

KANDEMİR, E., KANDEMİR, Y., 2012. Altın Eşanlamlı ve Karşıt Anlamlı Kelimeler Sözlüğü, Altın Kitaplar Yayınevi ve Ticaret A.Ş., 19. Basım, İstanbul.

KANTARDZİC, M., 2003. Data Mining:Concepts, Models, Methods and Algorithms, John Wiley&Sons.

KARADAĞ, A., TAKÇI, H., 2010. Metin Madenciliği ile Benzer Haber Tespiti, AB 2010, Akademik Bilişim, Muğla Üniversitesi, Muğla.

KAYA, M., & ÖZEL, S. 2014a. Integrating an online compiler and a plagiarism detection tool into the Moodle distance education system for easy assessment of programming assignments. Computer Applications in Engineering Education.

KAYA, M., 2011. Development of a Software on Distance Education Applications for Compilation and Plagiarism Detection of C Programming Language Assignments, MSc Thesis, Çukurova University, Institute of Natural and Applied Sciences, Department of Computer Engineering.

KAYA, M., 2012. Distance education systems used in universities of Turkey and Northern Cyprus. T. a. Proceedings of the 2nd World Conference on Learning (Dü.), Procedia - Social and Behavioral Sciences. içinde 31, s. 676–680. Elsevier.

KAYA, M., KELEŞ, A. E., LAPTALI-ORAL, E., 2013. Construction Crew Productivity Prediction by Using Data Mining Methods, Proceedings of the 4th World Conference on Learning, Teaching and Educational Leadership, Procedia - Social and Behavioral Sciences, 141, 1249–1253, 10.1016/j.sbspro.2014.05.215

KAYA, M., ÖZEL, S. A., 2014b, Türkçe Dokümanlardaki Benzerliklerin Tespiti İçin Mevcut Yazılımların Karşılaştırılması ve Türkçe Karakter Kullanımı ile Kök Almanın Etkisinin İncelenmesi, Çukurova Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 29(2), 115-129.

KELEŞ, A. E., KAYA, M., 2014. Duvar İnşa Edilmesinde Verimliliği Etkileyen Faktörlerin Apriori Veri Madenciliği Yöntemi Kullanılarak Analizi, Akademik Bilişim Konferansları, AB2014

KOOVAKKAI, D., 2011. Diagnosing Plague: Tools And Techniques For Detecting Plagiarism, 8th International CALIBER - 2011, Goa University, Goa, INFLIBNET Centre, Ahmedaba, India.

KOSALA, R., BLOCKEEL, H., 2000. Web Mining Research: A Survey, SIGKDD: SIGKDD explorations: newsletter of the special interest group (SIG) on knowledge discovery & data mining, ACM, 2(1): 1–15.

KOSKENNIEMI, K., 1983. Two-level morphology: a general computational model for word-form recognition and production. Publication No. 11. University of Helsinki: Department of General Linguistics.

KOTSIANTIS, S., KANELLOPOULOS, D., 2006. Association Rules Mining: A Recent Overview, GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), pp. 71-82.

KOZARIS, I. A., 2010. Platforms for e-learning, Analytical and Bioanalytical Chemistry, Springer-Verlag, Volume 397, Issue 3, pp 893-898.

KRAUSE, E. F., 1987. Taxicab Geometry. Dover. ISBN 0-486-25202-7.

KUMAR, R. K., CHADRASEKARAN, R. M., 2011. Attribute Correction-Data Cleaning Using Association Rule and Clustering Methods, International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.1, No.2, pp. 22-32, DOI : 10.5121/ijdkp.2011.1202.

KUSTANTO, C., LIEM, I., 2009. Automatic Source Code Plagiarism Detection. In Proceedings of the 2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, 481–486.

LANCASTER, T., CULWIN, F., 2000. A review of electronic services for plagiarism detection in student submissions, 8th Annual Conference on the Teaching of Computing, Edinburgh, http://www.ics.heacademy.ac.uk/events/presentations/317_Culwin.pdf.

LANE, P. C. R., LYON, C., MALCOLM, J. A., 2006. Demonstration of the Ferret plagiarism dectector. In Proceedings of the 2nd International Plagiarism Conference.

LEVENSHTEIN, V. I., 1966. Binary codes capable of correcting deletions, insertions, and reversals, Soviet Physics Doklady 10: 707–10.

LEWIS, G., 1953. Teach Yourself Turkish, English Universities Press, ISBN: 978-0-340-49231-4.

LEWIS, G., 2001. Turkish Grammar, Oxford University Press, ISBN: 0-19-870036-9.

LIAQAT, A. G., AIJAZ, A., 2011. Plagiarism Detection in Java Code, Linnaeus University, Faculty of Science and Engineering, School of Computer Science, Physics and Mathematics, Student thesis

LIN, C. Y., 2004. ROUGE: A Package for Automatic Evaluation of Summaries, Proc. of Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004, pp. 74-81, Barcelona, Spain.

LIU B., 2011. Web data mining: Exploring hyperlinks, contents, and usage data, Springer, ISBN 978-3-642-19460-3, pp. 624

LIU, B., HSU, W., & MA, Y., 1998. Integrating Classification and Association Rule Mining. Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining (KDD '98), pp. 80–86.

LONG-DISTANCE TEACHING, 1990. The Futurist, 24, 48

LOOMAN, J., CAMPBELL, J. B., 1960. Adaptation of Sorensen's K (1948) for estimating unit affinities in prairie vegetation, Ecology 41 (3): 409–416.

LOPRESTO, N., TURNER, A., 2000. Approximate String Matching, Proceedings of CSE 331 - Data Structures, University of Notre Dame, Department of Computer Science and Engineering, pp. 59-62.

LUKASHENKO, R., GRAUDINA, V., & GRUNDSPENKIS, J., 2007. Computer-Based Plagiarism Detection Methods and Tools: An Overview, International Conference on Computer Systems and Technologies (CompSysTech'07-Conference proceedings), University of Rousse, Bulgaria.

MADHULATHA, T. S., 2011. Comparison between K-Means and K-Medoids Clustering Algorithms, Advances in Computing and Information Technology Communications in Computer and Information Science, Volume 198, pp 472-481.

MAHESWARA RAO, V., VALLI KUMARI, V., & RAJU, K., 2010. Study of Visitor Behavior by Web Usage Mining, Information Processing and Management, Springer, pp. 181-187.

MALMI, L., KORHONEN, A., SAIKKONEN, R., 2002. Experiences in Automatic Assessment on Mass Courses and Issues for Designing Virtual Courses. In

Proceedings of the 7th Annual Conference on Innovation and Technology In Computer Science Education, page 55-59. New York, NY, USA, ACM.

MARQUES, J. M., LAZARO, D., JUAN, A. A., VILAJOSANA, X., DOMINGO, M., JORBA, J., 2013. PlanetLab@UOC: A Real Lab Over The Internet To Experiment With Distributed Systems. Computer Applications in Engineering Education, 21: 265–275.

MARTIN, F. C., 2009. Approximate string matching algorithms in art media archives, Master Thesis, AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics,

MAURER, H., KAPPE, F., & ZAKA, B. 2006. Plagiarism - A Survey. Journal of Universal Computer Science, 12(8), 1050-1084.

MAY, P., KLAU, G. W., BAUER, M., STEINKE, T., 2007. Accelerated microRNA-Precursor Detection Using the Smith-Waterman Algorithm on FPGAs, Distributed, High-Performance and Grid Computing in Computational Biology, Lecture Notes in Computer Science, Springer Berlin Heidelberg, Volume 4360, pp 19-32.

MENAI, M. E. B., 2012. Detection of Plagiarism in Arabic Documents, International Journal of Information Technology and Computer Science, 10, 80–89.

MERISOTIS, J. P., PHIPPS, R. A., The Institute For Higher Education Policy (IHEP), 2000. Quality on the line: Benchmarks for success in., IHEP Report, National Education Association (NEA), http://www.nea.org/assets/docs/HE/QualityOnTheLine.pdf

MOBASHER, B., 2006. Web Usage Mining, In John Wang (eds.), Encyclopedia of Data Warehousing and Mining, Idea Group.

MOHD, N., 2013. Clustering Learning Behavior of Students in Moodle using Association Rule Mining, pp. 1-4., Retrieved From: Acedemia.edu: https://ump.academia.edu/MohdNazzimanMohdMustapha/Papers.

MON, A. M., 2012. Spell checker for Myanmar language, 2012 International Conference on Information Retrieval & Knowledge Management (CAMP), pp.12,16, doi: 10.1109/InfRKM.2012.6204974

MONGE, A., ELKAN, C., 1996. The field matching problem: Algorithms and applications, In Proceedings of The Second International Conference on Knowledge Discovery and Data Mining, KDD.

MOODLE MANUALS. 2014. Retrieved from: http://docs.moodle.org/20/en/Moodle_manuals

MOODLE. 2014. Retrieved from: http://moodle.org/

MOSS PLAGIARISM MODULE, Access Date: 2013. https://moodle.org/plugins/view.php?plugin=plagiarism_moss

MOZGOVOY, M., 2006. Desktop tools for offline plagiarism detection in computer programs. Informatics in Education, 5(1), 97-112.

MOZGOVOY, M., KAKKONEN, T., & COSMA, G. 2010. Automatic Student Plagiarism Detection: Future perspectives. Journal of Educational Computing Research, 43(4), 507-527.

MUGRIDGE, I., 1991. Distance Education and The Teaching Of Science, Impact of Science on Society, 41, 4, 313-320

MUNDLURU, D., 2008. Automatically Constructing Wrappers for Effective and Efficient Web Information Extraction, Proquest LLC, Umi Microform, 167 pages.

MUSSO, G., 2007. Sequence Alignment (Needleman-Wunsch, Smith-Waterman), BCB410: Applied Bioinformatics Course – Lecture Notes, Retieved From: http://www.cs.utoronto.ca/~brudno/bcb410/lec2notes.pdf.

NAVARRO, G. and BAEZA-YATES, R., 1999. Very fast and simple approximate string matching, Information Processing Letters (IPL), 72: 65-70.

NEEDLEMAN, S. B., WUNSCH, C. D., 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins, Journal of Molecular Biology 48 (3): 443–53.

NEILL, C.J., SHANMUGANTHAN, G., 2004. A Web – enabled plagiarism detection tool, IT Professional, vol. 6, issue 5, pp. 19 – 23.

NORTH, D., 1993. Have video won't travel, Canadian Business, 68, 4, 43.

ÖZEN, Z., GÜLSEÇEN, S., 2012. Kaynak Kod Benzerliği ve Klon Kod Tespit Araçları, Akademik Bilişim'12 - XIV. Akademik Bilişim Konferansı Bildirileri Kitapçığı, Uşak.

ÖZSEVEN, T., DÜĞENCİ, M., 2011. LOG PreProcessing: Web Kullanım Madenciliği Ön İşlem Aşaması Uygulama Yazılımı, Akademik Bilişim 2011.

PANIGRAHI, N., 2015. Computing in Geographic Information Systems, CRC Press, 303 pages, ISBN: 978-1482223163.

PEDRAZA-PEREZ, R., ROMERO, C., & VENTURA, S., 2011. A Java desktop tool for mining Moodle data. in Pechenizkiy, M. et al. (Eds): 3rd Conference on Educational Data Mining, Eindhoven, The Netherlands: International EDM Society, pp. 319–320.

PLAGIARISM DETECT, http://www.plagiarism-detect.com/, Retrieved: 26.01.2013.

PLAGIARISMCHECKER, http://www.dustball.com/cs/plagiarism.checker/, Retrieved: 26.01.2013.

PLAGIARISMDETECT.COM, http://www.plagiarismdetect.com/, Retrieved: 26.01.2013.

PLAGIARISMDETECTION.ORG, 2014. http://www.plagiarismdetection.org, Retrieved: 2015-01-29.

PLAGIUM, http://www.plagium.com/, Retrieved: 26.01.2013.

PLAGSCAN, http://www.plagscan.com/, Retrieved: 26.01.2013.

PLONE, Access Date: 2013. http://plone.org/.

PORKODI, R., LSHIVAKUMAR, B., 2012. Rule based approach for constructing Gene/Protein names Dictionary from Medline abstract, International Journal of Advances in Computing and Information Technology, Volume 1, Issue 4, pp. 457-468.

PORTER, M., 1980. An Algorithm For Suffix Stripping, Program, Vol. 14, No. 3, pp. 130-137.

PORTER, M.F., 2001. Snowball: A Language for Stemming Algorithms. Available at: http://www.snowball.tartarus.org/texts/introduction.html.

POTENCIER, F., HUMPHREY, M., 2004-2008. Lingua::StopWords, Version 0.09, URL: http://search.cpan.org/~creamyg/Lingua-StopWords-0.09/lib/Lingua/StopWords.pm.

POTTHAST, M., STEIN, B., HAGEN, M., GOLLUB, T., BARRÓN-CEDEÑO, A., ROSSO, P., GUPTA, P., 2012. Pan12 detailed comparison training corpus, http://pan.webis.de/.

PRAISE, 2015. http://www.ics.heacademy.ac.uk/resources/assessment/plagiarism/detectiontools_comparison_source_free_text_comparison.html, Retrieved: 26.01.2015.

PRASAD, J. C., PANICKER, K. S. M., 2010. Single pattern search implementations in a cluster computing environment, 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST), pp.391,396, doi: 10.1109/DEST.2010.5610617.

PRECHELT, L., MALPOHL G., PHLIPPSEN, M., 2000. JPlag: Finding plagiarisms among a set of programs, Technical Report 2000-1, University of Karlsruhe, Germany.

PRECHELT, L., MALPOHL, G., PHILIPPSEN, M., 2002. Finding Plagiarisms among a Set of Programs with JPlag. Journal of Universal Computer Science (J.UCS), vol. 8, no. 11, pp. 1016 – 1038, DOI: 10.3217/jucs-008-11-1016.

RAJARAMAN, A., ULLMAN, J. D., 2011. Mining of Massive Datasets. Cambridge University Press, pp. 1–17, ISBN 978-1139058452.

RAMOS, J., TRENAS, M. A., GUTIÉRREZ, E., ROMERO, S., 2011. E-assessment of Matlab assignments in Moodle: Application to an introductory programming course for engineers. Computer Applications in Engineering Education, DOI: 10.1002/cae.20520

RANA, J. M. S., 2012. Bioinformatics - Tools & Applications, Uttarakhand State Biotechnology Department, Ministry of Science & Technology and Biotechnology, Government of Uttarakhand, ISBN: 978-8192329635.

REITER, E. and SOMAYAJULU, S., 2002. The SMART Retrieval System - Experiments in Automatic Document Processing, Computational Linguistics, 28/4, 447-485.

RIAD, A. M., FARAHAT, F. F., ASEM, A. S., & ZAHER, M. A. 2013. Studying Different Methods for Plagiarism Detection, International Journal of Computer Science and Engineering (IJCSE), 2, 147-154.

RIMER, S., 2003. A campus fad that's being copied: internet plagiarism seems on the rise, The New York Times, p. B7: http://www.nytimes.com/2003/09/03/nyregion/a-campus-fad-that-s-being-copied-internet-plagiarism-seems-on-the-rise.html

ROBERTS, D. W., 1986. Ordination on the basis of fuzzy set theory, Vegetatio, 66 (3): 123–131.

RODRIGUEZ, S., PEDRAZA, J. L., DOPICO, A. G., ROSALES, F., MENDEZ, R., 2007. Computer-based management environment for an assembly language programming laboratory. Computer Applications in Engineering Education, 15(1): 41-54, DOI 10.1002/cae.20094.

ROMERO, C., ESPEJO, P. G., ZAFRA, A., ROMERO, J. R. AND VENTURA, S. 2013. Web usage mining for predicting final marks of students that use Moodle courses. Comput. Appl. Eng. Educ., 21, 135–146. doi: 10.1002/cae.20456.

ROMERO, C., GONZÁLEZ, P., VENTURA, S., DEL JESUS, M. J., HERRERA, F., 2009. Evolutionary algorithms for subgroup discovery in e-learning: A practical application using Moodle data, Expert Systems with Applications, Volume 36, Issue 2, Part 1, pp. 1632-1644.

ROMERO, C., ROMERO, J. R., LUNA, J. M., VENTURA, S., 2010. Mining Rare Association Rules from e-Learning Data, International Conference on Educational Data Mining. Pittsburgh, PA, USA, pp. 171-180.

ROMERO, C., VENTURA, S., GARCÍA, E., 2008. Data mining in course management systems: Moodle case study and tutorial, Computers & Education, Volume 51, Issue 1, pp. 368-384, doi:10.1016/j.compedu.2007.05.016

RUSSELL, J., COHN, R., 2012. Taxicab Geometry, Book on Demand, 142 pages, ISBN: 978-5512171677.

SAFEASSIGN, http://safeassign.com/, Retrieved: 26.01.2013.

SALTON, G., 1971. The SMART retrieval system: Experiments processing, Englewood Cliffs, NJ: Prentice-Hall Inc.

SANDHYA, S., CHITRAKALA, S., 2011. Plagiarism Detection of Paraphrases in Text Documents with Document Retrieval, Advances in Computing and Information Technology, Communications in Computer and Information Science, Volume 198, pp. 330-338.

SCANLON, P. M., 2003. Student online plagiarism: how do we respond? College Teaching, 51, 4, 161-165.

SCHLEIMER, S., WILKERSON, D. S. and AIKEN, A., 2003. Winnowing: Local Algorithms For Document Fingerprinting, In Proceedings ACM SIGMOD International Conference On Management of Data, pp. 76-85.

SEESOURCES, http://www.plagscan.com/seesources/, Retrieved: 26.01.2013

SEVER, H., BITIRIM, Y., 2003. FindStem: Analysis and Evaluation of a Turkish Stemming Algorithm, String Processing and Information Retrieval, Lecture Notes in Computer Science, M.A. Nascimento, E.S.d. Moura and A.L. Oliveira (Eds.), Vol. 2857, pp. 238-251, Springer-Verlag, Heidelberg.

SHAW, M., 1980. Cheating policy in a Computer Science Department. SIGCSE Bulletin 12, 2, pp. 72-76.

SHCHERBININ, V., & BUTAKOV, S., 2008. Plagiarism Detection: The Tool and The Case Study, Proceedings of the IADIS International Conference on e-Learning, Amsterdam.

SHCHERBININ, V., & BUTAKOV, S., 2009. Using Microsoft SQL Server Platform for Plagiarism Detection, Proceedings of the 3rd PAN Workshop SEPLN PAN'09 (pp. 36-37). Spain: SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09).

SHERLOCK, 2015. http://sydney.edu.au/engineering/it/~scilect/sherlock/, Retrieved: 26.01.2015.

SHERLOCK-PLAGIARISM DETECTION SOFTWARE, 2014. http://www2.warwick.ac.uk/fac/sci/dcs/research/ias/software/sherlock/

SIM, 2014. http://dickgrune.com/Programs/similarity_tester/

SINGLA, N., GARG, D., 2012. String Matching Algorithms and their Applicability in various Applications, International Journal of Soft Computing and Engineering (IJSCE), Volume-I, Issue-6, pp. 218-222.

SMITH, T. F., WATERMAN, M. S., 1981a. Identification of Common Molecular Subsequences. Journal of Molecular Biology, 147, 195–197.

SMITH, T. F., WATERMAN, M. S., 1981b. Comparison of biosequences, Adv. Appl. Math., 2, 482–489.

SØRENSEN, T., 1948. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons, Kongelige Danske Videnskabernes Selskab 5 (4): 1–34.

SOURCE-CODE PLAGIARISM PLUGIN, Access Date: 2013. https://moodle.org/plugins/view.php?plugin=plagiarism_programming

SPINELLIS, D., ZAHARIAS, P., VRECHOPOULOS, A., 2007. Coping with plagiarism and grading load: Randomized programming assignments and reflective grading. Computer Applications in Engineering Education, 15(2): 113–123, DOI: 10.1002/cae.20096.

SPODICK, E., 1995. The Evolution of Distance Learning, Spode's Home Page: http://sqzm14.ust.hk/distance/

SRIVASTAVA, J., COOLEY, R., DESHPANDE, M., TAN, P. N., 2000. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data, ACM SIGKDD Explorations Newsletter, Volume: 1, Issue: 2, pp. 12-23.

STAMATATOS, E., 2011. Plagiarism Detection Based on Structural Information. Proc. of the 20th ACM Conference on Information and Knowledge Management (CIKM'11), pp. 1221-1230.

STOILOS, G., STAMOU, G., & KOLLIAS, S., 2005. A String Metric for Ontology Alignment. The Semantic Web – ISWC 2005 - 4th International Semantic Web Conference, Lecture Notes in Computer Sciences, 3729, s. 624-637.

SU, Z., AHN, B. R., EOM, K. Y., KANG, M. K., KIM, J. P., KIM, M. K., 2008. Plagiarism Detection Using the Levenshtein Distance and Smith-Waterman Algorithm, 3rd International Conference on Innovative Computing Information and Control, ICICIC '08., pp.569, doi: 10.1109/ICICIC.2008.422

SUNDERLAND-SMITH, W., 2008. Plagiarism, the Internet, and Student Learning: Improving Academic Integrity. New York: Routledge.

TAKALE, S. A., NANDGAONKAR, S. S., 2010. Measuring Semantic Similarity Between Words Using Web Documents", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 1, No.4.

TBD, TBD BİLİŞİM TERİMLERİ KARŞILIKLAR SÖZLÜĞÜ, http://www.tbd.org.tr/index.php?sayfa=sozluk&mi1. Retrieved: 12.07.2014.

TDK İMLA KILAVUZU, http://www.dilimiz.com/dil/imlakilavuzu/TDK/imlaanasayfasite.htm, Retrieved: 03.11.2011.

TDK, EŞ VE YAKIN ANLAMLI KELIMELER SÖZLÜĞÜ, http://www.tdk.gov.tr/index.php?option=com_esanlamlar&view=esanlamlar. Retrieved: 12.07.2014.

TDK, KISALTMALAR DIZINI, http://www.tdk.gov.tr/index.php?option=com_content&id=198:Kisaltmalar-Dizini&catid=50:yazm-kurallar. Retrieved: 12.07.2014.

TELEPOVSKÁ, H., GAJDOŠ, F., 2010. Plagiarism - Text Similarity Search, Electrical Engineering and Informatics Proceedings of the Faculty of Electrical Engineering and Informatics of the Technical University of Koşice, pp. 299-302.

THABTAH, F., COWLING, P., & HAMMOUD, S., 2006. Improving rule sorting, predictive accuracy and training time in associative classification. Expert Systems with Applications, 31, pp. 414–426.

THE COUNCIL OF WRITING PROGRAM ADMINISTRATORS, 2003. Defining and Avoiding Plagiarism: The WPA Statement on Best Practices, Retrieved: http://wpacouncil.org/files/wpa-plagiarism-statement.pdf.

THE GEORGETOWN UNIVERSITY HONOR COUNCIL, 1999. The Honor System, Retrieved From: https://bulletin.georgetown.edu/regulations/honor, Access Date: 03.07.2014.

THE UNIVERSITY OF SYDNEY TEACHING AND LEARNING COMMITTEE, 2003. Plagiarism detection software report, Draft One, http://sydney.edu.au/ab/committees/oldcom/learn_teach/agendas/2003/Plagiarism.pdf.

TOP, P., DOWLA, F., GANSEMER, J., 2007. A Dynamic Programming Algorithm for Name Matching, IEEE Symposium on Computational Intelligence and Data Mining, pp. 547-551.

TRESNAWATI, D., SYAICHU, R., KUSPRIYANTO, A., 2012. Plagiarism Detection System Design for Programming Assignment in Virtual Classroom based on Moodle, Procedia – Social and Behavioral Sciences, 67(10), 114-122, doi: 10.1016/jsbspro.2012.11.312.

TUNALI, V., & BİLGİN, T., 2012. Türkçe Metinlerin Kümelenmesinde Farklı Kök Bulma Yöntemlerinin Etkisinin Araştırılması. ELECO '2012 Elektrik - Elektronik ve Bilgisayar Mühendisliği Sempozyumu, pp. 598-602.

TUNALI, V., BILGIN, T. T., 2012. PRETO: A High-performance Text Mining Tool for Preprocessing Turkish Texts, International Conference on Computer Systems and Technologies (CompSysTech), Ruse, Bulgaria, June 22-23, 134-140.

TURKISH STOP WORD LIST 1.1, The Natural Language Processing Group, Department of Computer Engineering, Fatih University, nlp.ceng.fatih.edu.tr/blog/tr/?p=31, Retrieved: 25.12.2013.

TURNBAUGH, M. A., 2009. A Hybrid Template-based Composite Classification System, ProQuest, UMI Dissertation Publishing, 174 pages, ISBN: 978-1243605955.

TURNITIN PLUGIN, Access Date: 2013. https://moodle.org/plugins/view.php?plugin=plagiarism_turnitin

TURNITIN, http://www.turnitin.com/tr/us-categories/home/home, Retrieved: 26.01.2013.

UKKONEN, E., 1992. Approximate string-matching with q-grams and maximal matches, Theoretical Computer Science, 92, 191-211.

UNDERWOOD, J., & SZABO, A., 2003. Academic offences and e-learning: individual propensities in cheating. British Journal of Educational Technology, 34, 4, 467-477.

URKUND, 2015. https://moodle.org/plugins/view.php?plugin=plagiarism_urkund

VAST, 2015. http://www.ics.heacademy.ac.uk/resources/assessment/plagiarism/detectiontools_comparison_source_free_text_comparison.html, Retrieved: 26.07.2014.

VERIGUIDE, 2015. http://veriguide1.cse.cuhk.edu.hk/portal/plagiarism_detection/index.jsp, Retrieved: 26.07.2014.

VIKIPEDI, BILGISAYAR TERIMLERININ KISALTMALARININ LISTESI, http://tr.wikipedia.org/wiki/Bilgisayar_terimlerinin_k%C4%B1saltmalar%C4%B1n%C4%B1n_listesi, Retrieved: 27.02.2014.

VPL - VIRTUAL PROGRAMMING LAB, Access Date: 2013. https://moodle.org/mod/data/view.php?d=13&rid=4042&filter=1

WAGNER, N. R., 2010. Plagiarism by Student Programmers, The University of Texas at San Antonio. Division Computer Science. San Antonio, TX 78249, USA

WEN, J. R., NIE, J. Y. and ZHANG, H. J., 2002. Query clustering using user logs, ACM Transactions on Information Systems. 20/1 59-81.

WINKLER, W. E., 1990. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage, Proceedings of the Section on Survey Research Methods (American Statistical Association): 354–359.

WINKLER, W. E., 2006. Overview of Record Linkage and Current Research Directions, Research Report Series, RRS.

WISE, M. J., 1993b. Running Karp Rabin Matching and Greedy String Tiling, Technical Report Number 463, Dept. of CS, University of Sydney, http://www.it.usyd.edu.au/research/tr/tr463.pdf

WISE, M. J., 1995. Neweyes: A System for Comparing Biological Sequences Using the Running Karp-Rabin Greedy String-Tiling Algorithm, Third International Conference on Intelligent Systems for Molecular Biology Cambridge, England, July 16-19, pp 393–401.

WISE, M. J., 1996. YAP3: Improved Detection of Similarities in Computer Program and Other Texts, Proceedings of Twenty Seventh SIGCSE Technical Symposium on Computer Science Education, Philadelphia, USA. 130-134.

WISE, M.J., 1993a. String Similarity via Greedy String Tiling and Running Karp-Rabin Matching, Department of Computer Science, University of Sydney, (ftp://ftp.cs.su.oz.au/michaelw/doc/RKR_GST.ps) (Access Date: 01.07.2014)

WITTEN, H. I., FRANK, E., 2005. Data Mining Practical Machine Learning Tools and Technique, 2nd ed., Morgan Kaufmann Publishers, San Francisco, 560 pages.

WITTEN, I. H., FRANK, E., TRIGG, L., HALL, M., HOLMES, G., CUNNINGHAM, S.J., 1999. Weka: Practical machine learning tools and techniques with java implementations, In Proc. ICONIP/ANZIIS/ANNES'99 International Workshop: Emerging Knowledge In Engineering and Connectionist-Based Information Systems Dunedin, Hamilton, New Zealand, University of Waikato, Department of Computer Science, Working paper 99/11, pp. 192-196.

WORDNET, 2010. About WordNet. http://wordnet.princeton.edu/.

WWW-TRAKLA: A Web-Based System for Teaching Algorithms and Data Structures, Access Date: 2014, http://www.cs.hut.fi/~tred/WWW-TRAKLA/WWW-TRAKLA.html.

YENİTERZİ, R., 2011. Exploiting Morphology in Turkish Named Entity Recognition System, Proceedings of the ACL 2011 Student Session, Association for Computational Linguistics, p: 105-110.

YILMAZ, K. E., ARSLAN, A., YILMAZEL, Ö., 2009. Turkish Text Retrieval Experiments Using Lemur Toolkit, IADIS International Conference Applied Computing (IADIS AC), 2, 359-361.

YÜCESOY, B. and ÖĞÜDÜCÜ, S. G., 2007. Comparison of semantic and single term similarity measures for clustering Turkish documents, ICMLA ’07: Proceedings of the Sixth International Conference on Machine Learning and Applications. Washington, DC, USA: IEEE Computer Society, pp. 393–398.

YÜKSEL, M. E., TURNA, Ö. C., ERTÜRK, M. A., 2010. Bilgiye Erişim Sistemlerinde Veri Arama Ve Eşleştirme, Akademik Bilişim 2010, Muğla Üniversitesi.

ZEIDMAN, R. M., 2010. Detecting Plagiarism In Computer Source Code, United States Patent Application 20100325614 A1.

ZHANG, X., HEFLIN, J., 2010. Calculating Word Sense Probability Distributions for Semantic Web Applications, 2010 IEEE Fourth International Conference on Semantic Computing, Dept. of Computer Science and Engineering, Lehigh University PA 18015, USA.

ZOBEL, J., HAMILTON, M., 2002. Managing Student Plagiarism in Large Academic Departments, Australian University Review, 45, 2, 23-30.

CURRICULUM VITAE

Mümine KAYA was born in Adana, in 08.11.1986. She has completed her elementary education in Private Adana College. She went to intermediate school and high school at İsmail Safa Özler German Anatolian High School. After these education ended, she achieved Çukurova University Department of Computer Engineering of Engineering - Architecture Faculty in 2004. First year, she get English prep course in YADİM. Then she began to her undergraduate education in 2005. She has completed university education at Department of Computer Engineering of Engineering - Architecture Faculty of Çukurova University in 2009 and she has gotten Computer Engineering title. She began to her graduate education in 2009 Çukurova University Institute of Natural and Applied Sciences Department of Computer Engineering Computer Sciences Field. She got to work as a Research Assistant at Computer Engineering Department's Computer Sciences Field of Engineering Faculty of Osmaniye Korkut Ata University in Osmaniye in 2010. She achieved Çukurova University Institute of Natural and Applied Sciences Department of Computer Engineering Computer Sciences Field in 2011 and she began to her PhD education in 2011 Çukurova University Institute of Natural and Applied Sciences Department of Electrical and Electronics Engineering Computer Sciences Field. She resigned from her job in 2012 and got to work as a Research Assistant at Computer Engineering Department of Engineering and Natural Sciences Faculty of Adana Science and Technology University in Adana in 2012, and she has been working at the same department currently.

APPENDICES

Table 2.5. The comparison between the free textual plagiarism detection tools and web systems

	WCOPY FIND	COPYCA TCH GOLD	JPLA G	FERRET	SHERLO CK	SIM	DOCCOP	YTÜ DocumentSi milarity	YTÜ Text_Matc hing	PAIRw ise	Our Prop osal
Open source	Yes	No	No	No	Yes	Yes	No	Yes	Yes	Yes	Yes
Tool/Web System	Tool	Tool	Tool	Tool	Tool	Tool	Web	Tool	Tool	Web	Web
File Types	TXT, DOC	RTF, DOC, TXT, HTM	TXT, ASC, TEX	DOC, DOCX, RTF, ABW, TXT	TXT, TAR	DOC, DOCX, TXT, RTF	PDF DOC, DOCX	DOC, DOCX, TXT	TXT (Offline and Online Mode), DOC, HTML, PDF (Online Mode)	Most docume nt formats	TXT, DOC, DOC X, RTF, PDF,
File Size / # of files compared	500 Document s	320 Texts - 10,000 Words	NA	300 Document s - 10,000 Words	NA	NA	8 Files – Between 50 and 60000 Words	2 Files	NA	2 Files	No Size
Need Install	No	No	No	No	No	No	No	No	No	Yes	No
Local/Web-based	Local	Local	Web-based	Local	Local	Local	Web-based	Local	Both	Both	Web-based
Detect Added Words	Yes	Yes	NA	NA	NA	NA	NA	NA	NA	NA	Yes
Quick Response	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
Easy To Use	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Manual	Yes	Yes	No	Yes	No	Yes	No	Yes	Yes	Yes	Yes
Multiplatform	Yes	Yes	Yes	Yes	Yes	Yes	Web-Based	Yes	Yes	Web-Based	Yes
Usability	Excellent	Good	Excell ent	Excellent	Fair	Fair	Good	Excellent	Excellent	Good	Excell ent

Table 2.5. (Continued)

	WCOPY FIND	COPYCATCH GOLD	JPLAG	FERRET	SHERLOCK	SIM	DOCCOP	YTÜ DocumentSimilarity	YTÜ Text_Matching	PAIRwise	Our Proposal
Report	HTML Report	HTML Report, RTF Report, Side-By-Side Report	HTML Report, Side-By-Side Report	PDF Report, XML Report	Text File	Flat Text File	HTML Report with URL in E-mail	On interface display	On interface display	Side-By-Side Report, Graph Report	HTML Report
Presentation of Results	Good	Good	Excellent	Good	Fair	Fair	Excellent	Fair	Fair	Good	Excellent
LMS Integration	NA	NA	Yes	NA	Yes	Yes	NA	No	No	No	Yes
Language	41 languages	NA	NA	English, German	English	English	NA	Turkish	Turkish	English	Turkish
Turkish Language Supportes	Yes	No	No	No	No	No	No	Yes	Yes	No	Yes
Multiple Submitted Document Comparison	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Internet Checking	Yes but Google API licence required	Yes but Google API licence required	No	No	No	No	No	No	Yes	Yes	No
Synonym Checking	No	No	No	No	No	No	No	No	No	No	Yes
Sentence Structure Checking	No	No	No	No	No	No	No	No	No	No	No

Table 2.5. (Continued)

	WCOPY FIND	COPYCA TCH GOLD	JPLA G	FERRET	SHERLO CK	SIM	DOCCOP	YTÜ DocumentSi milarity	YTÜ Text_Matc hing	PAIRw ise	Our Prop osal
Translation Checking	No	No	No	No	No	No	No	No	No	No	Yes (Engli sh to Turki sh for IT field)
Verbatim/ Cut-Paste Checking	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Paraphrase Checking	No	No	No	No	No	No	No	No	No	No	No
Image/ Multimedia Checking	No	No	No	No	No	No	No	No	No	No	No
Reference Validity Checking	No	No	No	No	No	No	No	No	No	No	No
Interface	Graphical	Graphical	Graphi cal	Graphical	Command Line	Comman d Line	Web Page	Graphical	Graphical	Web Page	Web Page
Operating Systems	Windows	Windows	Windo ws	Windows, Unix/Linu x, Mac OS	Windows, Unix/Linu x	Windows , Unix/Lin ux	Web-Based	Windows, Unix/Linux	Windows, Unix/Linux	Windo ws	Wind ows, Unix/ Linux
Written in	C++	Java	Java	C++	C	C	NA	Java	NA	NA	Perl, Java, PHP

Table 2.5. (Continued)

	WCOPY FIND	**COPYCATCH GOLD**	**JPLAG**	**FERRET**	**SHER LOCK**	**SIM**	**DOCCOP**	**YTÜ Document Similarity**	**YTÜ Text Matching**	**PAIR wise**	**Our Proposal**
Algorithm	Word-based string Matching	String matching	Greedy string tiling algorithm	Word-3-gram Fingerprinting	Word-n-gram Fingerprinting (digital signatures)	dynamic programming string alignment technique	NA	Cosine similarity, Jaccard similarity coefficient, Dice coefficient, Tanimoto coefficient, n-Word gram	Cosine Similarity	Identical word usage	Greedy String Tiling Algorithm, Cosine Similarity

NA: Not Available

Table 2.6. The comparison between the commercial textual plagiarism detection tools and web systems

	COPYCATCH GOLD	TURNITIN	ITHENTICATE	MyDropBox	Docol©c	EVE2	GLATT	Our Proposal
Open source	No	No	No	No	No	No	No	Yes
Tool/Web System	Tool	Web	Web	Web	Web	Tool	Tool	Web
File Types	RTF, DOC, TXT, HTM	DOC, DOCX, PDF, HTML, RTF, TXT, PS, PPT, PPTX, PPS, HWP, ODT	DOC, DOCX, XML, TXT, PDF, HTML, WPD, RTF, ODT	ZIP, DOC, TXT, PDF,RTF, HTML	PDF, DOC, DOCX, RTF, HTML, PPT,PPTX, XLS, XLSX,TXT, ODT	NA	NA	TXT, DOC, DOCX, RTF, PDF
File Size / # of files compared	320 Texts - 10,000 Words	20 MB - 400 Pages	40 MB – 400 Pages - 25,000 words	NA	NA	NA	NA	No Size
Local/Web-based	Local	Web-based	Web-based	Web-based	Web-based	Web-based	Neither	Web-based
Need Install	No	No	No	No	No	Yes	NA	No
Detect Added Words	Yes	Yes	Yes	NA	NA	NA	NA	Yes
Quick Response	Yes	Yes	Yes	No	NA	No	NA	No
Easy To Use	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Manual	Yes	Yes	Yes	NA	Yes	Yes	Yes	
Report	HTML Report, RTF Report, Side-By-Side Report	HTML Report	HTML Report	NA	E-mail	HTML Report	NA	HTML Report
Multiplatform	Yes	Web-Based	Web-Based	Web-Based	Web-Based	No	NA	Yes
Usability	Good	Good	Good	Good	Good	Good	Good	Excellent
Presentation of Results	Good	Good	Good	Good	Good	Good	Good	Excellent
Language	NA	21 languages	30 languages	NA	NA	NA	English	Turkish

Table 2.6. (Continued)

	COPYCATCH GOLD	TURNITIN	ITHENTICATE	MyDropBox	Docol©c	EVE2	GLATT	Our Proposal
Turkish Language Supportes	No	Yes	Yes	No	No	No	No	Yes
Multiple Submitted Document Comparison	Yes	Yes but only with its database	Yes but only with its database	NA	NA	No	No	Yes
Internet Checking	Yes but Google API licence required	Yes	Yes	Yes	Yes	Yes	NA	No
Synonym Checking	No	No	No	No	No	No	No	Yes
Sentence Structure Checking	No	Yes	No	No	No	No	No	No
Verbatim/Cut-Paste Checking	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Paraphrase Checking	No	No	No	Yes	No	No	No	No
Translation Checking	No	No	No	No	No	No	No	Yes (English to Turkish for IT field)
Image/Multimedia Checking	No	No	No	No	No	No	No	No
Reference Validity Checking	No	No	No	No	No	No	No	No
LMS Integration	NA	Yes	Yes	Yes	NA	No	No	Yes
Interface	Graphical	Graphical	Graphical	Web Page	Web Page	NA	NA	Web Page
Operating Systems	Windows	Web-Based	Web-Based	NA	NA	Windows	NA	Windows, Unix/Linux
Written in	Java	NA	NA	NA	NA	NA	NA	Perl, Java, PHP

Table 2.6. (Continued)

	COPYCATCH GOLD	TURNITIN	ITHENTICATE	MyDropBox	Docol©c	EVE2	GLATT	Our Proposal
Algorithm	String matching	Comparison to web and archiving the former submissions	Comparison to web and archiving the former submissions	A highly advanced algorithm	Google API	NA	Fingerprint method	Greedy String Tiling, Cosine Similarity

NA: Not Available

Table 4.10. Three Texts Created From The YTÜ Kemik Web Site

The First Text
Hande Ataizi gizlice evlendi Hande Ataizi 1.5 yıldır birlikte olduğu sevgilisi avukat Fethi Pekin ile dün Ritz Carlton Oteli'nde aile arasında yapılan nikahla evlendi. Otelin residance bölümünde Pekin'in kendisine ait olan dairede 19.30'daki kıyılan nikaha gazeteciler alınmadı. Aile arasında alınan kararla basına hiçbir şekilde görüntü ya da fotoğraf verilmeyeceği açıklandı. Fethi Pekin'in babası Ahmet Pekin oğlunun Hande Ataizi ile evlenmesine karşı çıkıyordu
The Second Text
Nikâh şahidinin evinde kalıyor Fethi Pekin ile evliliğini, kayınpederi Ahmet Pekin yüzünden 24 saat sonra bitiren Hande Ataizi, nikâh şahidinin evine sığındı. Hande, balayına giderken eşiyle kavga etmiş, onu havaalanında bırakıp Süzer Plaza'daki evlerine dönmüştü. Ancak birkaç gün sonra kesin olarak boşanmaya karar verince nikâh şahitliğini yapan yakın arkadaşı İmregül Gencer'in evine taşındı. Bu arada Fethi Pekin'in de The Ritz Carlton'un residence bölümündeki dairesi yerine bir otelde kaldığını duydum. Pekin, Hande ile 1.5 yıl birlikte yaşadıkları dairede, Hande'den izler olduğu için içeri adım atmak istemiyormuş! Bu yüzden ayrılığın ardından dairesinde tadilat başlatan Pekin, evdeki tüm eşyaları da değiştirmiş. Yarın dairesine geçecek olan Pekin, Hande'ye dair tüm izleri hayatından silip atmak istiyormuş. Bu, o kadar kolay mı acaba?
The Third Text
Geçen hafta evlenen Hande Ataizi ile Fethi Pekin, balayına giderken kavga edip ayrıldılar. Göcek'e balayına gitmek için havaalanına doğru yola çıkan çift, yolda büyük bir kavgaya tutuştu. Hande de bavulunu alıp evine döndü. Hande Ataizi, 1,5 yıldır flört ettiği ve "Çok seviyorum" dediği avukat sevgilisi Fethi Pekin'le 4 Ağustos'ta nikâh masasına oturup muradına ermişti. Nikâh, çiftin birlikte oturduğu The Ritz Carlton'un rezidans bölümündeki dairelerinde sesiz sedasız gerçekleşmişti. İlişkilerinin başladığı günden itibaren medya ile çok fazla görüşmeyen, fotoğraf vermeyen çiftin nikâhı da aynı gizlilik içinde olmuştu. Hande Ataizi, düğünden bir süre önce yaptığı bir röportajında, "Fethi işlerinden dolayı çok fazla medyanın gözü önünde olmak istemiyor. Benden de bazı şeylerin aramızda kalmasını, basının duymamasını rica etti. Onun bu ricasına saygı duyuyorum ve çok fazla konuşmak istemiyorum. Herkes gibi sıradan bir şekilde evleneceğim. Abartı hiçbir şey yok" şeklinde konuşmuştu. BABA PEKİN İSTEMEDİ! Ancak sosyete ve sanat çevrelerinde konuşulanlara göre bu gizliliğin asıl nedeni baba Ahmet Pekin'di. İlişkinin çok fazla medyaya yansımasından rahatsız olan baba Ahmet Pekin, nikâhın basına sızmadan gizlilik içinde yapılmasını özellikle istemişti. Bu durum, uzun süredir bu mutlu günü bekleyen ve nikâhında giymek için Christian Dior marka bir de gelinlik alan Hande'yi üzse de sevgilisini kırmayarak bu isteği kabul etmişti. Bir süre önce Mandarin Otelleri ve Orangina'nın sahibi işadamı Jean Pierre Bansard'ın kızının St. Tropez'deki düğün töreninde katılan bir arkadaşımın anlattığına göre, baba Ahmet Pekin ile Hande Ataizi düğünde sıkı bir kavgaya tutuşmuş, araya giren Fethi Pekin ortamı yumuşatmış. Baba Pekin, kesinlikle düğün istemediğini, ancak sade bir nikâhla evlenebileceklerini ilk kez St. Tropez'deki düğünde dile getirmiş. NEDEN, MAL AYRILIĞI SÖZLEŞMESİ Mİ? Çiçeği burnunda evlilerin ayrılmasına neden olarak, baba Ahmet Pekin'in "Evlilikte mal ayrılığı" sözleşmesi istemesinden kaynaklandığı iddia ediliyor. Konuşulanlara göre nikâh günü çiftin dairesine gelen Ahmet Pekin, Hande Ataizi'ne imzalaması için bu sözleşmeyi uzatıyor. Ve bunu imzalamadan evlenemeyeceklerini belirtiyor. Hande ise avukatına göstermeden hiçbir şeyi imzalamayacağını söylüyor. Baba Pekin bu duruma çok kızsa da yine oğlu Fethi araya girip, ortalığı sakinleştiriyor. Ve Hande sözleşmeye imza atmadan, nikâh defterine imzayı atmayı başarıyor! Ancak düğünün ertesi günü, Ali Koç'un Göcek'te demirli olan teknesinde geçirecekleri balayı tatili için havaalanına doğru yola çıkan çift, sohbet ederlerken, laf dönüp dolaşıp bu sözleşmeye geliyor. Ve orada çıngar çıkıyor! Bu yüzden sıkı bir kavgaya tutuşan yeni evliler, havalimanına vardıklarında yollarını ayırıyor. Fethi Pekin Dalaman uçağına binerken, gözü yaşlı Hande de eve geri dönüyor! BARIŞACAKLAR MI, YOKSA... Fethi Pekin ile Hande Ataizi'nin ancak 24 saat süren mutluluklarının akıbetinin ne olacağı tam bir merak konusu. Hande'yi deliler gibi seven ve onunla evlenmek için her türlü fedakarlığı göze alan Fethi Pekin, bakalım Hande'nin gönlünü alıp, barışacak mı? Ya da çift, sadece 24 saat süren evliliklerini bitirip, boşanacaklar mı? Nikâh günü, The Ritz Carlton'un rezidans bölümündeki dairelerini, Hollanda'dan özel olarak getirtilen ve "beraberliğin başlangıcı" anlamına gelen 850 beyaz gülle donatan çiftin mutluluğunun bu kadar kısa sürmesi herkesi şaşkına çevirmiş durumda.

Table 4.12. The Experimental Results for Determining the Distance Metrics After The Preprocessing Phases

Distance Metrics	**No Preprocessing**			**Removing Stopword and Turkish Characters**			**Removing Stopword and Turkish Characters, Using Zemberek**		
	1-2	**1-3**	**2-3**	**1-2**	**1-3**	**2-3**	**1-2**	**1-3**	**2-3**
Jaro	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Levenshtein	0.26	0.11	0.3	0.3	0.13	0.18	0.29	0.13	0.19
EuclideanDistance	0.90	0.93	0.17	0.88	0.93	0.93	0.89	0.91	0.92
JaccardSimilarity	0.12	0.06	0.09	0.08	0.03	0.05	0. 2	0.13	0.16
QGramsDistance	0.35	0.19	0.3	**0.41**	0.21	0.32	0.39	0.21	0.30
CosineSimilarity	0.44	0.45	0.56	0.60	0.52	0.55	**0.61**	**0.61**	**0.62**
JaroWinkler	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
BlockDistance	0.17	0.09	0.14	0.18	0.1	0.11	0.33	0.19	0.24
ChapmanLengthDeviation	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ChapmanMatchingSoundex	0.72	0.84	0.79	0.71	0.82	0.77	0.85	0.94	0.88
ChapmanMeanLength	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
ChapmanOrderedNameCompoundSimilarity	0.24	0.27	0.25	0.30	0.26	0.26	0.32	0.27	0.29
MatchingCoefficient	0.14	0.05	0.09	0.18	0.06	0.07	0.29	0.11	0.15
MongeElkan	0.91	0.94	0.92	0.86	0.92	0.9	0.95	0.98	0.97
NeedlemanWunch	0.54	0.51	0.52	0.56	0.51	0.52	**0.57**	**0.51**	**0.52**
OverlapCoefficient	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SmithWaterman	0.05	0.06	0.05	0.06	0.11	0.06	0.09	0.17	0.07
SmithWatermanGotoh	0.14	0.20	0.15	0.23	0.31	0.22	0.25	0.36	0.22
SmithWatermanGotohWindowedAffine	0.06	0.05	0.03	0.09	0.08	0.05	0.12	0.11	0.07
Soundex	0.0	0.33	0.0	0.0	0.0	0.0	0.0	0.0	0.0
[TagLink_[similarity metrics.TagLinkToken@305f6ed4]	0.50	0.48	0.51	0.54	0.49	0.5	**0.60**	**0.53**	**0.53**
[TagLinkToken_Tr_0.3]	0.14	0.1	0.11	0.2	0.12	0.13	0.20	0.14	0.15

Table 4.13. Two Texts Created From The Homeworks

The First Text
-Bubble short algoratmasındaki takas işlemleri zor yapıldıgı için selection short bubble short algoritmasının geliştirilmiş algoritmasıdır. -En iyi durum verilen dizinin küçükten büyüğe sıralı gelmiş halidir.Dizi içinde en kucuk eleman bulunur,diger alt dizilerde yapılacak karsılastırma işlemlerinde for döngüsüne girilecek ama if döngüsüne dizi sıralı oldugu için girmeyecektir,bu sonuc T(n) sonucunu azaltacak fakat O(n) karesel artıs gösterecektir. -En kötü durum ise verilecek dizi büyükten küçüğe sıralı olup en küçük eleman eleman n kere karsılastırılıp alt dizide n-1 karsılastırma yapıp T(n) artıracaktır fakat yine en büyük kat sayılı degişkeni aldıgımızda O(n) yine karesel olacaktır.
The Second Text
kabarcık algoritmasının dedeğişim işlemleri çok zor olduğu için seçme sıralaması, kabarcık sıralaması algoritmasının geliştirilmiş algoritmasıdır. en iyi durum bize verilen dizinin bize sıralı gelmiş olmasıdır, dizi bize sıralı geldiği için for döngüsüne girer ancak if kısmına sıralı olduğu için girmeyecektir. sonuc olarak T(n) sonucu azaltacak 0(n) bize artışın karesel olduğunu gösterecektir. en kötü durumda ise dizinin bize tersten sıralı geldmiş olduğu haldir. bu durumda diziyi n kere karşılaştırılmış olur ve alt dizi de ise n-1 kez karsılaştıma yapılır böylece t(n) artmış olur ama buna karşın en büyük kat sayılı değişkeni alırsak o(n) yine karesel olacaktır.

Table 4.18. The used assignments in the experiment of Section 4.5

File_1
Soru 3. Bir SSD diskin yapısı nasıldır? Nasıl çalışır? Veriler fiziksel olarak nasıl saklanır? Araştırınız. Solid State Disk'ler aslında kullandığımız usb flash belleklerin birbirlerine bağlanarak oluşturulduğu bir platformdur. Tabi bunu üzerindeki kontrolcü sayesinde kusursuz olarak gerçekleştirir ve bir bütün olarak çalışır. Örneğin 64GB'lık bir SSD diskte 32 adet 2GB'lık flash bellek veya 128GB'lıkta 32 adet 4GB'lık flash bellek kullanılabilir.Flash bellekler EEPROM çiplerin bir çeşididir. Aşağıda veri iletimi ve depolamanın nasıl yapıldığını şematik olarak görmektesiniz: Mor renkli kutu olarak görülmekte olanlar kontrol ve kayar nokta mekanizmalarını içeren transistörlerdir. Bu iki transistör ince bir oksit tabakasıyla birbirinden ayrılmıştır. Kayar nokta kapısı(floating gate) sadece kontrol kapısından(control gate) geçen kelime çizgisine(wordline) bağlıdır. Bu bağlantı aşamasında hücre 1 değerini alır, bunu 0 yapabilmek için Fowler-Nordheim Tunneling adı verilen işlem basamağının gerçekleşmesi gerekir. Tunneling işlemi, elektronların kayar nokta kapısındaki hareketini tanımlar. Yaklaşık 10 ile 13 volt arasında gelen elektrik akımı, yüküyle beraber bit hattından(bitline) gelir ve kayar nokta kapısından girerek önce sürekli akım kanalına(drain) daha sonra da kaynak kanalına(source) geçerek topraklanır.Bu yüklenme kayar nokta kapısı transistörünün bir elektron tabancası gibi hareket etmesine neden olur. Uyarılmış bu elektronlar ince oksit tabakasının diğer tarafına itilir ve negatif yüklenirler. Bu negatif yüklü elektronlar kontrol ve kayar nokta kapıları arasında bir bariyer vazifesi görürler. Hücre sensörü isimli özel bir yorumlayıcı bölüm bulunur. Bu bölüm sayesinde kayar nokta kapısından geçen yük miktarı izlenir. Eğer bu kapıdan yükün %50'sinden fazlası geçiyorsa bunun değeri 1?dir. Eğer yük miktarı %50?nin altında kalıyorsa da değeri 0 olarak tanımlanır. Flash bellek çiplerinin içinde yer alan hücrelerdeki elektronlarlar normale dönüp 1 değerini alırlar. Bunu da yüksek gerilim yükü ile oluşan elektrik alanı ile sağlarlar. Flash bellekler kapalı devredirler ve süregelen döngüsel görevlerin uygulanması şeklinde çalışırlar. Eğer bir veri silinecekse, daha önceden belirlenmiş bloklara gereken elektrik alan uygulanır ve blok sıfırlanmış olur. Silinen kısma tekrar yazılması mümkündür. Flash bellekler geleneksel EEPROM'lardan çok daha hızlıdırlar. Çünkü EEPROM'lar birim zamanda tek byte'lık veri temizleyebilirken, flash belleklerde bu bloklar halinde temizlenerek çok daha hızlı silinir ve tekrar yazılırlar.
File_2
Soru 3. Bir SSD diskin yapısı nasıldır? Nasıl çalışır? Veriler fiziksel olarak nasıl saklanır? Araştırınız. Solid-State Disk (SSD) nedir? Geleneksel sabit disk sürücülerin yerine kullanılması amaçlanan, NAND flash tabanlı bir depolama çözümüdür. Samsung tarafından geliştirilen bu teknoloji, veri depolama alanında yepyeni bir dönem başlatmaktadır. Bu yeni nesil depolama çözümü; içerisinde hareketli manyetik disklerin bulunduğu sabit disklere oranla, daha düşük güç tüketimi, yüksek dayanıklılık, hafiflik ve performans artışı gibi özelliklere sahiptir. Olağanüstü Okuma-Yazma ve Yükleme Performansı Bigboy SSD'nin performansı sıradan sabit disklere göre %150 daha fazladır. Örneğin, veri okuma hızı saniyede 57MB ve veri yazma hızı saniyede 32MB değerlerindedir, bu da sabit disklerin iki katından daha fazladır. Diğer bir deyişle SSD, 1.8 inçlik bir sabit diske göre bilgisayar açılma süresini yarıya indirmektedir. Depolama cihazı olan SSD, motor, disk ve okuyucu kafa gibi hareketli parçalar yoktur. Bu sayede HDDlerdeki çalışma süresi(spin-up time), arama süresi (seek time) ve yüksek hızda veri aktarımı sırasında yaşanan gecikme süreleri gibi değerleri ortadan kaldırmaktadır. SSD kullanıldığı cihazlar için dost bir cihazdır. Hareketli parçası olmaması gürültüsüz çalışmasını sağlar ve çok düşük düzeyde olan güç tüketimi ısınmasını önler. Taşınabilir Bilgisayarlar ile Tam Uyumlu SSD'yi sıradan bir hard disk şeklinde tasarlanmıştır. Ara yüzü ATA5 özellikleri ile tam uyumludur ve 0-4 PIO modları ile UDMA mode 4 (Ultra DMA 66MHz) desteğine sahiptir. Düşük Güç Gereksinimi Hareketli parça bulundurmaz

Çalışma ve bekleme sırasında çok düşük güç harcaması
Pil kullanım süresinde %10 artış
Hız performansı
Açılma süresi yok
Arama süresi yok
Devir gecikmesi yok
Artırılmış veri aktarım hızı
57MB/saniye okuma, 32MB/saniye yazma hızları
İki kat hızlı açılan sistem
Güvenilir
Bozulabilecek hareketli parça bulundurmaz
Yüksek darbe ve titreşim dayanımı
Uç sıcaklık değerlerinde çalışabilme (-20° - 80°C aralığında)
Gelişmiş çalışma biçimi
Gürültüsüz
Isınmaz
Sıradan sabit disklerin yarısı ağırlığında
Teknik Özellikler :
Hız: 58MB/sn Okuma, 32MB/sn Yazma
Arayüz: ATA-5
Gerilim: 3,3V - 5V
Akustik (Çalışma/Boşta): 0 dB
Güç: Çalışma 0,5W, Boşta 0,1W, Bekleme 0,06W
Ağırlık: 50g
Başka bir kaynaktan
SSD Diskler Nedir ? Ve SSD' ler Hakkında Bilinen 10 Yanlış
SSD (Solid State Disk), önümüzdeki kısa bir süreç içerisinde mekanik disklerin yerini alacak veri depolama aygıtlarıdır. Günümüzde yaygın olarak kullanmış olduğumuz mekanik diskler üzerinde performansı artırmak için RPM (revolutions per minute) değeri artırılmış fakat bununla birlikte sıcaklık problemi ile karşılaşılmıştır. Mekanik diskler ile ilgili problemlerin çözümü ve yeni geliştirmelerin hemen hemen son noktaya yaklaştığını söylememiz yanlış olmayacaktır. Mekanik diskler artık yerini kendisinden performans ve dayanıklılık anlamında çok daha gelişmiş bir teknoloji olan SSD disklere bırakmaya başlamıştır.
SSD teknolojisi içerisinde mekanik disklerde olduğu gibi dönen bir mekanizma söz konusu değildir. Hal böyle olunca mekanik disklere göre daha az narindir. Mekanik diskler sarsılmalar sonucunda daha yüksek oranda problem oluştururken, SSD diskler çok daha dayanıklı bir yapıya sahiptir. Hareketli parçaların kullanılmaması aynı zamanda daha sessiz çalışmalarını sağlamaktadır.
SSD'ler sahip oldukları depolama yongalarına göre iki çeşittirler.
Flash yonga tabanlı
SDRAM yonga tabanlı SSD'ler.
SSD'ler 1", 1.8" ve 2.5" boyutlarının dışında kurumsal veri depolama ihtiyaçları için disk üniteleri (disk storage) olarak da üretilmektedir. Kurumsal yapılar artık, yüksek kapasiteli verilerini çok daha güvenilirli ve hızlı bir yapıya sahip olan SSD Storage'lar üzerinde depolayabilecektir.
Solid State Sürücüler (SSD) henüz çok yeni oldukları için özellikle kurumsal alanda kullanımları ile ilgili birçok yanlış kanı mevcut. Bu yazıda, en yaygın 10 kavram hakkında doğruları bulabilirsiniz.
1- SSD'ler Çok Pahalı
Gerçek : SSD'lerin ilk yatırım maliyetleri yüksek görünmesine rağmen toplam sahip olma maliyetine (TCO) bakıldığında geleneksel sabit disk sürücülerinden çok daha ucuz olduğu görülebilir. Çok pahalı dediğimiz ancak geleneksel sabit disklerden teorik olarak 1000 kat daha hızlı erişim sürelerine sahip Solid State Sürücüler kullanıldığında 50-60 adet sunucunun kümelenmesi yerine çok daha az sunucu ile çok daha yüksek performans elde edilmektedir. Geleneksel sabit diskler kullanıldığında ihtiyaç duyulan sunucu ve disk sayısındaki bu azalmaya paralel olarak, bu disk dizinlerini ve sunucuları çalıştırmak ve soğutmak için kullanılan enerji tüketimindeki azalma (%80 daha az), her bir sunucu için ödenen işletim sistemi ve uygulama yazılımlarının lisans maliyetlerindeki azalma, bakım onarım anlaşmaları maliyetlerindeki azalma,

sunucuların ve disklerin kapladığı alandaki azalma, acil durum için çalıştırılan personel sayısındaki azalma göz önüne alındığında, SSD'lerin toplam sahip olma maliyetlerinin çok daha düşük olduğu ve bilgi işlem merkezlerine her yıl milyonlarca lira tasarruf sağladığı görülmektedir.

2- NAND Flash Tabanlı SLC (Tek Katmanlı Hücre) veya MLC (Çok Katmanlı Hücre) SSD'ler güvenli değil

Gerçek: Genel olarak bakıldığında hem SLC hem de MLC NAND Flash tabanlı diskler sabit disklerden çok daha güvenilirdir. Çünkü mekanik parçaları yoktur. Hem SSD'ler hem de HDD'ler hata düzeltme kodu (ECC) kullanmaktadır. Her iki tür sürücü içinde hata düzeltme gereksinimi, yaklaşık 12.5 terabayt okumada birdir. Ayrıca, SSD'lerin titreşim ve elektro manyetik parazitlere olan dayanıklılığı HDD'lerle ölçülemeyecek derecede yüksektir. SSD'lerin 1990'ların ortalarından beri askeri ve kritik uygulamalarda kullanılmasının nedeni bu sağlamlık ve güvenilirlikleridir.

3- Hacker'lar Solid State Sürücüleri Sabit Sürücüler'den Daha Hızlı Hackler'ler

Gerçek: Hem HDD, hem SSD'ler hackerlar tarafından kırılabilir. Bilgisayarların korunma seviyesi güvenlik yazılımlarıyla ilgilidir. SSD'lerin HDD'lere göre teorik olarak 1000 kat daha hızlı bir şekilde veriyi getirebilmesi göz önüne alındığında bu durum gayet normal, ancak SSD'lerin sadece hacker'lar için tasarlandığını düşünmek ve bunu SSD'lerin bir zayıflığı olarak göstermek doğru değildir.

4- Özellikle çok veri yazıldığında SLC veya MLC Flash NAND tabanlı SSD'lerin ömrü kısadır

Gerçek: Flash SSD'lerdeki her bir hücrenin yazma dayanıklılık ömrü günümüzde 2 milyon döngüdür, yani tek bir hücre üzerine güvenli olarak 2 milyon defa yazılabilir. SSD'lerde bir bilginin yazılması için herhangi bir yazma kafasına veya döner bir bileşene ihtiyaç yoktur. Yazılacak hücre adreslenir ve doğrudan o adresteki hücre üzerine silme-yazma işlemi yapılır. Yazma ömrünü uzatmak için aynı hücre üzerine defalarca yazılmaz, bunun yerine sunucuyu ilgilendirmeyen bir algoritma kullanılarak bilgiler milyarlarca hücre arasından istatistikî olarak en az kullanılan hücreler üzerine yazılır. 64 GB'lık tek bir Flash SSD kullandığımızı (bir dizinde daha fazla ve yüksek kapasiteli sürücülerin kullanılması çalışma ömrünü uzatacaktır) ve yazma hızımızın 80Mbytes/s olduğunu varsayarsak (ki bu günümüzdeki en hızlı flash SSDlerin ortalama yazma hızıdır) 64 GB'lık bir SSD flash diskin her bir hücresinin üzerine minimum yazma-dayanıklılığı olan 2.000.000 defa silip yazmamız 55 yılımızı alacaktır. Buda sadece bilgi teknolojisi için değil, biz insanlar için bile oldukça uzun bir süredir.

5- SSD'lerin kapasiteleri çok düşük

Gerçek: Kurumsal-sınıf SSD aygıtı üreticisi olan Violin Memory Inc. 2U'luk bir şasi içinde optimize edilmiş RAID4 özelliklerine haiz 8 TB kullanılabilir kapasite sunabiliyor. 19" bir rack kabin içinde bunlardan 16 tanesi olduğu düşünülürse, tek bir kabinde 128 TB'lık bir kapasite elde edilebilmektedir. SSD'lerin kapasitesi sadece sunuculardaki yüklü olan işletim sisteminin adresleme kapasitesi ile sınırlıdır.

6- SSDlerin veri koruma ve kurtarma özellikleri çok zayıf

Gerçek: Kullanılan veri koruma yazılımlarının haricinde, ayrı disk dizinlerine ve RAID kontrolörlerine ihtiyaç duyan HDD ve diğer SSD sistemlerinin aksine, Violin1010'nun mimari yapısını oluşturan Violin Anahtarlanmış Bellek teknolojisi, maksimum veri koruma performans, yoğunluk ve enerji verimi için RAID Kontrolörünü bünyesinde bütünleşik olarak barındırmaktadır. 16KB'lik veri 4K'lık bloklar halinde ayrılmakta ve her bir 4K'lık veri farklı RAID grupları üzerindeki farklı modüllere 1K olarak yazılmakta, pariteleri ise RAID grubunu teşkil eden 5. modül üzerinde tutulmaktadır. Bu mimari ve RAID algoritması verinin en üst seviyede korunmasını sağlamaktadır. Violin 1010 cihazı içindeki 84 modülden herhangi ikisi arızalandığında bile, önce dört adet canlı yedek modüllerden biri otomatik olarak arızalanan modülün yerini almakta ve RAID algoritması ile korunan veri, önce canlı yedek modüllerden herhangi biri üzerine aktarılmakta, RAID grubu birkaç dakika içinde (HDD'lerden çok daha hızlı olarak) sistemdeki herhangi bir yedek evrensel modül üzerine otomatik olarak yeniden yapılanmakta ve yapılandırılmadan sonra, arızalı modüller sistem ve programlar çalışırken çıkartılarak yeni modüller ile değiştirilebilmekte (hot swapping) ve canlı yedek modül tekrar eski işlevine geri dönmektedir. Hem HDD'ler hem de SSD'ler işletim sistemi tarafından fiziksel bir depolama cihazı olarak görülmektedir. Bu nedenle bilgi işlem merkezinin kullandığı hata giderme yazılımına bağlı olarak, HDD'ler kullanıldığında saatlerce süren veri kurtarma işlemi SSD'ler

kullanıldığında dakikalar içinde tamamlanabilmektedir.

7- Bütün SSD'ler aynı performans ve güvenilirliğe sahip

Gerçek: Bütün SSD cihazları eşit olarak yaratılmamışlardır. Bir disk kontrol biriminin alınıp, bir NAND flash ile aynı kutuya koyulması başarılı bir SSD elde edilmesini sağlamaz. SSD mimarisinin bir bütün olarak ele alınması gerekir. Örneğin, kullanılan disk kontrol birimi, NAND teknolojisi ve genel SSD mimarisinin güvenilirlik, ölçeklilik, dayanıklılık, yıpranma dengeleme ve performans üzerinde önemli etkileri vardır. Ancak patenti Violin Memory Inc'e ait olan anahtarlanmış bellek mimarisi incelendiğinde, bu mimarinin neden çok yüksek sürdürebilir "350K read" ve "220K write" performansı gösterdiği, kullanıcılara donanıma yerleşik RAID4 özelliğinin yanı sıra, nasıl daha yüksek kullanılabilir kapasite sağlayabildiği açıkça görülebilir.

8- SSD'lerin sağladığı enerji tasarrufu o kadar önemli değil

Gerçek : Solid State Sürücülerde, kullanılan bellek modülleri için önemli olan parametre her GByte için ihtiyaç duyulan enerjidir. Bu nedenle enerji verimi, bilgi işlem merkezlerindeki sistemlerin toplam sahip olma maliyetini düşürmek için en önemli parametredir. Genel olarak, 3-5 yıllık ömürleri boyunca, donanımların enerji maliyetleri yatırım maliyetlerinin üstüne çıkmaktadır. Sadece dizüstü bilgisayarlar ve kişisel bilgisayarlar düşünülse bile, HDD'lerle kıyaslandığında "read" işlemleri için SSD'lerin harcadığı enerji yaklaşık %50, "write işlemleri için %30 daha azdır. Bir bilgi işlem merkezinde kullanılan disk dizinleri ve bunların bağlı olduğu sunucular düşünüldüğünde, gerek duyulan sunucu ve disk sayısındaki azalma nedeniyle elde edilen enerji tasarrufu oldukça önemlidir. Buna birde artık gerek duyulmayan bu sunucuların ve HDD'lerin soğutulması için harcanan enerji tüketimi eklendiğinde elde edilen tasarruf %80 seviyelerine çıkmaktadır.

9- HDD veriyi SSD'lerden daha uzun bir süre güvenli şekilde depolayabilir.

Gerçek: HDD'lerin write döngüsü açısından belirlenmiş herhangi bir dayanıklılık limiti olmamasına rağmen, bozulma oranları daha yüksek olup, aniden arızalanabilirler. SSD'lerin dayanıklılığının veya güvenilirliğinin kullanıcılar için bir endişe teşkil etmesi gerekmemektedir. Asıl endişe edilmesi gereken durumun bir HDD veya bir SSD dizini arızalandığında, verilerin ne kadar sürede geri yüklenebileceği ve buna bağlı olarak üretimde ne kadarlık bir kayba uğranacağı olmalıdır. HDD'lerin çalışma ömrü genelde 3-5 yıldır. Violin Memory Inc. Türkiye'de kendi SSD ürünleri için hem dayanıklılık hem de veri depolama özellikleri için en az 5 yıllık bir garanti vermektedir.

10- SSD'ler çok kısa bir süre içinde HDD'lerin yerini alacaktır

Gerçek: SSD'lerin performans, toplam sahip olma maliyeti ve güvenirlilik açısından çok büyük avantajları olmasına rağmen, yeni teknolojiye geçişte karşılaşılan bir çok gereksiz endişe, psikolojik ve ihtiyati nedenlerle daha uzun bir süre HDD'ler ile bir arada var olacağı beklenmektedir. Ancak IDC verilerine göre silikon depolama pazarı inanılmaz bir büyüme göstermektedir. Birkaç yıl içinde silikon depolamanın çok yaygın olarak kullanılacağı beklenmektedir.

Sonuç olarak yukarıda bahsi geçen noktalarda dikkate alındığında SSD disklerin Bilgi İşlem sistemleri için ne kadar önemli olduğu ve birçok kazanım sağladığı açıkça görülmektedir. SSD ile ilgili avantajları bir kez daha özetlemek gerekirse aşağıdaki maddeleri dile getirebiliriz.

- Çok daha yüksek IO performansı
- Hareketli parça olmamasından dolayı daha sessiz çalışma
- Çok daha az enerji tüketimi ve daha az ısınma
- Hareketli parça olmamasından dolayı daha fazla dayanıklılık
- Fiziksel olarak daha az alan kaplama
- Okuma ve yazma işlemlerinde daha az hata
- İşletme maliyetlerinin daha düşük olması
- Daha düşük Toplam Sahip Olma (TCO) maliyeti"

File_3

Solid State Disk'ler aslında kullandığımız usb flas belleklerin birbirlerine bağlanarak oluşturulduğu bir platformdur. Tabi bunu üzerindeki kontrolcü sayesinde kusursuz olarak gerçekleştirir ve bir bütün olarak çalışır. Örneğin 64GB'lık bir SSD diskte 32 adet 2GB'lık flash bellek veya 128GB'lıkta 32 adet 4GB'lık flash bellek kullanılabilir. Bunun bir örneğini sağda içi açılmış SSD kartı üzerinde görmektesiniz.

Nasıl Çalışır
Flash bellekler EEPROM çiplerin bir çeşididir. Aşağıda veri iletimi ve depolamanın nasıl yapıldığını şematik olarak görmektesiniz:

Mor renkli kutu olarak görülmekte olanlar kontrol ve kayar nokta mekanizmalarını içeren transistörlerdir. Bu iki transistör ince bir oksit tabakasıyla birbirinden ayrılmıştır. Kayar nokta kapısı(floating gate) sadece kontrol kapısından(control gate) geçen kelime çizgisine(wordline) bağlıdır. Bu bağlantı aşamasında hücre 1 değerini alır, bunu 0 yapabilmek için Fowler-Nordheim Tunneling adı verilen işlem basamağının gerçekleşmesi gerekir.
Tunneling işlemi, elektronların kayar nokta kapısındaki hareketini tanımlar. Yaklaşık 10 ile 13 volt arasında gelen elektrik akımı, yüküyle beraber bit hattından(bitline) gelir ve kayar nokta kapısından girerek önce sürekli akım kanalına(drain) daha sonra da kaynak kanalına(source) geçerek topraklanır.
Bu yüklenme kayar nokta kapısı transistörünün bir elektron tabancası gibi hareket etmesine neden olur. Uyarılmış bu elektronlar ince oksit tabakasının diğer tarafına itilir ve negatif yüklenirler. Bu negatif yüklü elektronlar kontrol ve kayar nokta kapıları arasında bir bariyer vazifesi görürler. Hücre sensörü isimli özel bir yorumlayıcı bölüm bulunur. Bu bölüm sayesinde kayar nokta kapısından geçen yük miktarı izlenir. Eğer bu kapıdan yükün %50'sinden fazlası geçiyorsa bunun değeri 1?dir. Eğer yük miktarı %50?nin altında kalıyorsa da değeri 0 olarak tanımlanır.
Flash bellek çiplerinin içinde yer alan hücrelerdeki elektronlarlar normale dönüp 1 değerini alırlar. Bunu da yüksek gerilim yükü ile oluşan elektrik alanı ile sağlarlar. Flash bellekler kapalı devredirler ve süregelen döngüsel görevlerin uygulanması şeklinde çalışırlar. Eğer bir veri silinecekse, daha önceden belirlenmiş bloklara gereken elektrik alan uygulanır ve blok sıfırlanmış olur. Silinen kısma tekrar yazılması mümkündür. Flash bellekler geleneksel EEPROM'lardan çok daha hızlıdırlar. Çünkü EEPROM'lar birim zamanda tek byte'lık veri temizleyebilirken, flash belleklerde bu bloklar halinde temizlenerek çok daha hızlı silinir ve tekrar yazılırlar.

File_4

Soru 3. Bir SSD diskin yapısı nasıldır? Nasıl çalışır? Veriler fiziksel olarak nasıl saklanır? Araştırınız.
SSDD (Solid State Disk Drive veya Katı Hâl Disk Sürücü): veri depolamak için geliştirilmiş hard-disk'lerin yerini alan veri depolama aygıtıdır. Mekanik bir harddiskin yazma hızı ortalama 200mb/sn iken SSDD'lerde bu hız 1gb/sn'dir. En büyük avantajları...ısı, ses ve mekanikliğin ortadan kalkmasıdır.
Yapısı: Solid State Disk'ler aslında kullandığımız usb flash belleklerin birbirlerine bağlanarak oluşturulduğu bir platformdur. Örneğin 64GB'lık bir SSD diskte 32 adet 2GB'lık flash bellek veya 128GB'lıkta 32 adet 4GB'lık flash bellek kullanılabilir. Bunun bir örneğini aşağıda içi açılmış SSD kartı üzerinde görmektesiniz.
Nasıl Çalışır? Veriler fiziksel olarak nasıl saklanır?
Ssdlerin nasıl çalıştığını ve fiziksel olarak verilerin nerde saklandığını anlamamız için flash belleklerin çalışma prensibini incelememiz gerekir.
Flash bellekler EEPROM çiplerin bir çeşididir. Aşağıda veri iletimi ve depolamanın nasıl yapıldığını şematik olarak görmektesiniz:
Mor renkli kutu olarak görülmekte olanlar kontrol ve kayar nokta mekanizmalarını içeren transistörlerdir. Bu iki transistör ince bir oksit tabakasıyla birbirinden ayrılmıştır. Kayar nokta kapısı(floating gate) sadece kontrol kapısından(control gate) geçen kelime çizgisine(wordline) bağlıdır. Bu bağlantı aşamasında hücre 1 değerini alır, bunu 0 yapabilmek için Fowler-Nordheim Tunneling adı verilen işlem basamağının gerçekleşmesi gerekir.
Tunneling işlemi, elektronların kayar nokta kapısındaki hareketini tanımlar. Yaklaşık 10 ile 13 volt arasında gelen elektrik akımı, yüküyle beraber bit hattından(bitline) gelir ve kayar nokta kapısından girerek önce sürekli akım kanalına(drain) daha sonra da kaynak kanalına(source) geçerek topraklanır.
Bu yüklenme kayar nokta kapısı transistörünün bir elektron tabancası gibi hareket etmesine neden olur. Uyarılmış bu elektronlar ince oksit tabakasının diğer tarafına itilir ve negatif yüklenirler. Bu negatif yüklü elektronlar kontrol ve kayar nokta kapıları arasında bir bariyer vazifesi görürler. Hücre sensörü isimli özel bir yorumlayıcı bölüm bulunur. Bu bölüm sayesinde kayar nokta kapısından geçen yük miktarı izlenir. Eğer bu kapıdan yükün %50'sinden fazlası geçiyorsa bunun

değeri 1?dir. Eğer yük miktarı %50?nin altında kalıyorsa da değeri 0 olarak tanımlanır.
Flash bellek çiplerinin içinde yer alan hücrelerdeki elektronlarlar normale dönüp 1 değerini alırlar. Bunu da yüksek gerilim yükü ile oluşan elektrik alanı ile sağlarlar. Flash bellekler kapalı devredirler ve süregelen döngüsel görevlerin uygulanması şeklinde çalışırlar. Eğer bir veri silinecekse, daha önceden belirlenmiş bloklara gereken elektrik alan uygulanır ve blok sıfırlanmış olur. Silinen kısma tekrar yazılması mümkündür. Flash bellekler geleneksel EEPROM'lardan çok daha hızlıdırlar. Çünkü EEPROM'lar birim zamanda tek byte'lık veri temizleyebilirken, flash belleklerde bu bloklar halinde temizlenerek çok daha hızlı silinir ve tekrar yazılırlar.
Veriler fiziksel olarak SSD içerisindeki flash belleklerin içindeki bloklarda saklanırlar…

File_5

Cevap 3.
Bir Solid State Disk (SSD) in temel unsurları, tamamen elektronik devreler üzerinde yer alan çipli sistemlerdir. SSD nin temel yapısını oluşturan 4 temel unsur vardır.
Bunlar: 1- Baskı Devre Kartı
2- Bağlantı sağlayan bir arayüz (SATA veya PCI-E)
3- İşlemci (SSD Kontrolcüsü)
4- NAND flash bellek çipleri SSD lerin çalışma mantığı HDD'lerden oldukça farklıdır. Geleneksel harddisklerde veriler plakaların üzerindeki sektörlerde tutulur. Bu plakalar hard disk içerisindeki motorlarla döndürülüp, kafa kısmı ile okunurken; SSD lerde veriler NAND-flash yongalarda saklanır. SSD lerin içindeki bellek yongaları, tıpkı bilgisayarlardaki RAM'ler gibi hızlıdır. SSD ler yapılarında EEPROM lar taşırlar. SSD lerdeki flash yongalar RAM lere göre biraz daha yavaş olmakla beraber daha uzun süre veri depolayabilirler. Elektrik akımı kesilse bile verilerde kayıp yaşanmaz

File_6

Soru 3. Bir SSD diskin yapısı nasıldır? Nasıl çalışır? Veriler fiziksel olarak nasıl saklanır? Araştırınız.
Cevap 3.
Bir Solid State Disk (SSD) in temel unsurları, tamamen elektronik devreler üzerinde yer alan çipli sistemlerdir.
SSD nin temel yapısını oluşturan 4 temel unsur vardır. Bunlar:
Baskı Devre Kartı
Bağlantı sağlayan bir arayüz (SATA veya PCI-E)
İşlemci (SSD Kontrolcüsü)
NAND flash bellek çipleri
SSD lerin çalışma mantığı HDD'lerden oldukça farklıdır. Geleneksel harddisklerde veriler plakaların üzerindeki sektörlerde tutulur. Bu plakalar hard disk içerisindeki motorlarla döndürülüp, kafa kısmı ile okunurken; SSD lerde veriler NAND-flash yongalarda saklanır. SSD lerin içindeki bellek yongaları, tıpkı bilgisayarlardaki RAM'ler gibi hızlıdır. SSD ler yapılarında EEPROM lar taşırlar.
SSD lerdeki flash yongalar RAM lere göre biraz daha yavaş olmakla beraber daha uzun süre veri depolayabilirler. Elektrik akımı kesilse bile verilerde kayıp yaşanmaz

Table 4.35. The used assignments in the experiment of Section 4.9

File1
Same as File1 in Table 4.18.
File2
------ CEVAP 3 ------ SSD'nin YAPISI SSD üniteleri marka, model, form faktör, kapasite gibi pek çok değişken yüzünden birbirinden farklı yapılarda olabilir. Ancak bir SSD ünitesi temel olarak alttaki elemanlardan oluşur; 1-PCB (Baskı Devre Kartı):Bütün elemanları üzerinde barındıran ve onların iletişimlerini sağlayan elemandır. 2-Bilgisayara bağlantıyı sağlayan bir ara yüz(SATA veya PCI-E):SSD ünitelerinde şu anda 2 çeşit ara yüz kullanılıyor: SATA ve PCI-E (PCI Express) PCI-E daha hızlı bir ara yüz ve sadece masaüstü bilgisayarlarda kullanılabiliyor. SATA ara yüzü ise an itibari ile piyasada bulunan modellerde SATA 3.0 GBit/s (SATA 2) ve SATA 6.0 GBit/s (SATA 3) olarak 2 tipte bulunuyor. SATA 2 ara yüzü 280 MB/s veri aktarma kapasitesine sahipken, bu aktarım hızı SATA 3 ara yüzünde 550 MB/s. 3-NAND flash bellek çipleri:SSD ünitesinin en küçük yapı elemanı NAND-flash hücrelerdir. Bir NAND-flash hücre, 1 yada 2 byte veri saklayabilir.Eğer 1 byte depolayabiliyorsa Single Level Cell (SLC), 2 byte depolayabiliyorsa Multi Level Cell (MLC) adı verilir. SLC tipi NAND-flash hücrelere sahip SSD üniteleri daha uzun kullanım süresi, daha yüksek veri yazma hızları ve sistem dosyası gibi küçük boyutlu çok sayıda dosyayı daha iyi işleme yeteneklerine sahiptir. 4-SSD CONTROLLER:Tüm SSD üniteleri, yazma ve okuma işlemini düzenleyen ve bilgisayarlarda kullandığımız işlemcilere benzer hesaplama kabiliyetleri olan kontrolörlere sahiptir. Bu kontrolör SSD ünitesini SSD yapan şeydir ve performansına direkt etki eder. Performansı etkileyen şeyler ise çok çeşitlidir: aynı anda yazılabilecek veri miktarı, yazma hızı, bellek bant genişliği ve bellek verimliliği bunların başında gelir. İşte tüm bunları yöneten ve performansı belirleyen bellek kontrolörüdür. Kontrolörün en önemli olduğu konu ise performansa çok önemli etkisi olan "Write Amplification" fenomenidir. Write Amplification, SSD kontrolcüsünün yazmak zorunda olduğu veri miktarı ile işletim sisteminin yazmak istediği veri miktarı arasındaki orandır. Bu değer standart bir SSD ünitesi için 1'in üstündedir. Yani 1 MB boyutunda veri yazmak istediğinizde SSD ünitesi bunu yapabilmek için 1 MB'tan fazla miktarda yazma işlemi yapmak zorundadır. SSD' İN ÇALIŞMA MANTIĞI Mor renkli kutu olarak görülmekte olanlar kontrol ve kayar nokta mekanizmalarını içeren transistörlerdir. Bu iki transistör ince bir oksit tabakasıyla birbirinden ayrılmıştır. Kayar nokta kapısı (floating gate) sadece kontrol kapısından (control gate) geçen kelime çizgisine (wordline) bağlıdır. Bu bağlantı aşamasında hücre 1 değerini alır, bunu 0 yapabilmek için Fowler-Nordheim Tunneling adı verilen işlem basamağının gerçekleşmesi gerekir. Tunneling işlemi, elektronların kayar nokta kapısındaki hareketini tanımlar. Yaklaşık 10 ile 13 volt arasında gelen elektrik akımı, yüküyle beraber bit hattından (bitline) gelir ve kayar nokta kapısından girerek önce sürekli akım kanalına (drain) daha sonra da kaynak kanalına (source) geçerek topraklanır. Bu yüklenme kayar nokta kapısı transistörünün bir elektron tabancası gibi hareket etmesine neden olur. Uyarılmış bu elektronlar ince oksit tabakasının diğer tarafına itilir ve negatif yüklenirler. Bu negatif yüklü elektronlar kontrol ve kayar nokta kapıları arasında bir bariyer vazifesi görürler. Hücre sensörü isimli özel bir yorumlayıcı bölüm bulunur. Bu bölüm sayesinde kayar nokta kapısından geçen yük miktarı izlenir. Eğer bu kapıdan yükün %50'sinden fazlası geçiyorsa bunun değeri 1?dir. Eğer yük miktarı %50?nin altında kalıyorsa da değeri 0 olarak tanımlanır. Flash bellek çiplerinin içinde yer alan hücrelerdeki elektronlar normale dönüp 1 değerini alırlar. Bunu da yüksek gerilim yükü ile oluşan elektrik alanı ile sağlarlar. Flash bellekler kapalı devredirler ve süregelen döngüsel görevlerin uygulanması şeklinde çalışırlar. Eğer bir veri silinecekse, daha önceden belirlenmiş bloklara gereken elektrik alan uygulanır ve blok sıfırlanmış olur. Silinen kısma tekrar yazılması mümkündür. Flash bellekler geleneksel EEPROM'lardan çok daha hızlıdırlar. Çünkü EEPROM'lar birim zamanda tek byte'lık veri temizleyebilirken, flash

belleklerde bu bloklar halinde temizlenerek çok daha hızlı silinir ve tekrar yazılırlar.

FİZİKSEL SAKLAMA

SSD bellek türünde bilgilerin yazılması için ve silinmesi için normal elektrik voltajı uygulanır. Diğer ROM türlerinde olduğu gibi içerdiği veriyi elektrik kesilse de saklar. Veriler sabit bloklar halinde yazılır.

SSD bellekler de hücrelerden oluşur. Her hücrenin kendi transistörleri vardır. Bilgisayar otamında bilgiler 0 ve 1'lerden oluşur. 0'lar düşük voltaj, 1'ler ise yüksek voltaj anlamına gelir. Veri yazılmak istendiği anda, transistörlerin voltaj seviyeleri değiştirilerek bilgiler yazılır silinir yenilenir.

Transistörlerin voltaj ile uyarılması ile "0" ve "1" değerleri oluşur. Sonuçta her bir hücre "0" ya da "1" değerleri ile doldurulur ve bu değerlerin birleşmesi ile de veriler meydana gelir.Buraya kadar her şey standart RAM modüllerindekiyle birebir aynıdır. Flash bellekleri standart RAM modüllerinden ayıran ise, yazılan verilerin güç kesintisinde bile silinmemesi.

File3

CEVAP 3

SSD DİSKİN YAPISI

SSD üniteleri marka, model, form faktör, kapasite gibi pek çok değişken yüzünden birbirinden farklı yapılarda olabilir. Ancak bir SSD ünitesi temel olarak ŞU şu elemanlardan oluşur;

1-PCB (Baskı Devre Kartı):Bütün elemanları üzerinde barındıran ve onların iletişimlerini sağlayan elemandır.

2-Bilgisayara bağlantıyı sağlayan bir ara yüz(SATA veya PCI-E):SSD ünitelerinde şu anda 2 çeşit ara yüz kullanılıyor: SATA ve PCI-E (PCI Express) PCI-E daha hızlı bir ara yüz ve sadece masaüstü bilgisayarlarda kullanılabiliyor. SATA ara yüzü ise an itibari ile piyasada bulunan modellerde SATA 3.0 GBit/s (SATA 2) ve SATA 6.0 GBit/s (SATA 3) olarak 2 tipte bulunuyor. SATA 2 ara yüzü 280 MB/s veri aktarma kapasitesine sahipken, bu aktarım hızı SATA 3 ara yüzünde 550 MB/s.

3-İşlemci:Tüm SSD üniteleri, yazma ve okuma işlemini düzenleyen ve bilgisayarlarda kullandığımız işlemcilere benzer hesaplama kabiliyetleri olan kontrolörlere sahiptir. Bu kontrolör SSD ünitesini SSD yapan şeydir ve performansına direkt etki eder. Performansı etkileyen şeyler ise çok çeşitlidir: aynı anda yazılabilecek veri miktarı, yazma hızı, bellek bant genişliği ve bellek verimliliği bunların başında gelir. İşte tüm bunları yöneten ve performansı belirleyen bellek kontrolörüdür.

Kontrolörün en önemli olduğu konu ise performansa çok önemli etkisi olan "Write Amplification" fenomenidir. Write Amplification, SSD kontrolcüsünün yazmak zorunda olduğu veri miktarı ile işletim sisteminin yazmak istediği veri miktarı arasındaki orandır. Bu değer standart bir SSD ünitesi için 1'in üstündedir. Yani 1 MB boyutunda veri yazmak istediğinizde SSD ünitesi bunu yapabilmek için 1 MB'tan fazla miktarda yazma işlemi yapmak zorundadır.

4-NAND flash bellek çipleri:SSD ünitesinin en küçük yapı elemanı NAND-flash hücrelerdir. Bir NAND-flash hücre, 1 yada 2 byte veri saklayabilir.Eğer 1 byte depolayabiliyorsa Single Level Cell (SLC), 2 byte depolayabiliyorsa Multi Level Cell (MLC) adı verilir. SLC tipi NAND-flash hücrelere sahip SSD üniteleri daha uzun kullanım süresi, daha yüksek veri yazma hızları ve sistem dosyası gibi küçük boyutlu çok sayıda dosyayı daha iyi işleme yeteneklerine sahiptir.

SSD' İN ÇALIŞMA MANTIĞI

Mor renkli kutu olarak görülmekte olanlar kontrol ve kayar nokta mekanizmalarını içeren transistörlerdir. Bu iki transistör ince bir oksit tabakasıyla birbirinden ayrılmıştır. Kayar nokta kapısı (floating gate) sadece kontrol kapısından (control gate) geçen kelime çizgisine (wordline) bağlıdır. Bu bağlantı aşamasında hücre 1 değerini alır, bunu 0 yapabilmek için Fowler-Nordheim Tunneling adı verilen işlem basamağının gerçekleşmesi gerekir.

Tunneling işlemi, elektronların kayar nokta kapısındaki hareketini tanımlar. Yaklaşık 10 ile 13 volt arasında gelen elektrik akımı, yüküyle beraber bit hattından (bitline) gelir ve kayar nokta kapısından girerek önce sürekli akım kanalına (drain) daha sonra da kaynak kanalına (source) geçerek topraklanır.

Bu yüklenme kayar nokta kapısı transistörünün bir elektron tabancası gibi hareket etmesine neden olur. Uyarılmış bu elektronlar ince oksit tabakasının diğer tarafına itilir ve negatif yüklenirler. Bu negatif yüklü elektronlar kontrol ve kayar nokta kapıları arasında bir bariyer

vazifesi görürler. Hücre sensörü isimli özel bir yorumlayıcı bölüm bulunur. Bu bölüm sayesinde kayar nokta kapısından geçen yük miktarı izlenir. Eğer bu kapıdan yükün %50'sinden fazlası geçiyorsa bunun değeri 1?dir. Eğer yük miktarı %50?nin altında kalıyorsa da değeri 0 olarak tanımlanır.

Flash bellek çiplerinin içinde yer alan hücrelerdeki elektronlar normale dönüp 1 değerini alırlar. Bunu da yüksek gerilim yükü ile oluşan elektrik alanı ile sağlarlar. Flash bellekler kapalı devredirler ve süregelen döngüsel görevlerin uygulanması şeklinde çalışırlar. Eğer bir veri silinecekse, daha önceden belirlenmiş bloklara gereken elektrik alan uygulanır ve blok sıfırlanmış olur. Silinen kısma tekrar yazılması mümkündür. Flash bellekler geleneksel EEPROM'lardan çok daha hızlıdırlar. Çünkü EEPROM'lar birim zamanda tek byte'lık veri temizleyebilirken, flash belleklerde bu bloklar halinde temizlenerek çok daha hızlı silinir ve tekrar yazılırlar.

VERİLERİN FİZİKSEL OLARAK KAYDEDİLMESİ

SSD bellek türünde bilgilerin yazılması için ve silinmesi için normal elektrik voltajı uygulanır. Diğer ROM türlerinde olduğu gibi içerdiği veriyi elektrik kesilse de saklar. Veriler sabit bloklar halinde yazılır

SSD bellekler de hücrelerden oluşur. Her hücrenin kendi transistörleri vardır. Bilgisayar otamında bilgiler 0 ve 1'lerden oluşur. 0'lar düşük voltaj, 1'ler ise yüksek voltaj anlamına gelir. Veri yazılmak istendiği anda, transistörlerin voltaj seviyeleri değiştirilerek bilgiler yazılır silinir yenilenir.

Transistörlerin voltaj ile uyarılması ile "0" ve "1" değerleri oluşur. Sonuçta her bir hücre "0" ya da "1" değerleri ile doldurulur ve bu değerlerin birleşmesi ile de veriler meydana gelir.Buraya kadar her şey standart RAM modüllerindekiyle birebir aynıdır. Flash bellekleri standart RAM modüllerinden ayıran ise, yazılan verilerin güç kesintisinde bile silinmemesi.

File4

Cevap 3-)

SSD (Solid State Disk), önümüzdeki kısa bir süreç içerisinde mekanik disklerin yerini alacak veri depolama aygıtlarıdır. Günümüzde yaygın olarak kullanmış olduğumuz mekanik diskler üzerinde performansı artırmak için RPM (revolutions per minute) değeri artırılmış fakat bununla birlikte sıcaklık problemi ile karşılaşılmıştır. Mekanik diskler ile ilgili problemlerin çözümü ve yeni geliştirmelerin hemen hemen son noktaya yaklaştığını söylememiz yanlış olmayacaktır. Mekanik diskler artık yerini kendisinden performans ve dayanıklılık anlamında çok daha gelişmiş bir teknoloji olan SSD disklere bırakmaya başlamıştır.

SSD teknolojisi içerisinde mekanik disklerde olduğu gibi dönen bir mekanizma söz konusu değildir. Hal böyle olunca mekanik disklere göre daha az narindir. Mekanik diskler sarsılmalar sonucunda daha yüksek oranda problem oluştururken, SSD diskler çok daha dayanıklı bir yapıya sahiptir. Hareketli parçaların kullanılmaması aynı zamanda daha sessiz çalışmalarını sağlamaktadır.

SSD'ler sahip oldukları depolama yongalarına göre iki çeşittirler.

Flash yonga tabanlı

SDRAM yonga tabanlı SSD'ler.

SSD'ler 1", 1.8" ve 2.5" boyutlarının dışında kurumsal veri depolama ihtiyaçları için disk üniteleri (disk storage) olarak da üretilmektedir. Kurumsal yapılar artık, yüksek kapasiteli verilerini çok daha güvenilirli ve hızlı bir yapıya sahip olan SSD Storage'lar üzerinde depolayabilecektir.

Solid State Sürücüler (SSD) henüz çok yeni oldukları için özellikle kurumsal alanda kullanımları ile ilgili birçok yanlış kanı mevcut. Bu yazıda, en yaygın 10 kavram hakkında doğruları bulabilirsiniz.

1- SSD'ler Çok Pahalı

Gerçek : SSD'lerin ilk yatırım maliyetleri yüksek görünmesine rağmen toplam sahip olma maliyetine (TCO) bakıldığında geleneksel sabit disk sürücülerinden çok daha ucuz olduğu görülebilir. Çok pahalı dediğimiz ancak geleneksel sabit disklerden teorik olarak 1000 kat daha hızlı erişim sürelerine sahip Solid State Sürücüler kullanıldığında 50-60 adet sunucunun kümelenmesi yerine çok daha az sunucu ile çok daha yüksek performans elde edilmektedir. Geleneksel sabit diskler kullanıldığında ihtiyaç duyulan sunucu ve disk sayısındaki bu azalmaya paralel olarak, bu disk dizinlerini ve sunucuları çalıştırmak ve soğutmak için kullanılan enerji tüketimindeki azalma (%80 daha az), her bir sunucu için ödenen işletim sistemi ve uygulama yazılımlarının lisans maliyetlerindeki azalma, bakım onarım anlaşmaları maliyetlerindeki azalma,

sunucuların ve disklerin kapladığı alandaki azalma, acil durum için çalıştırılan personel sayısındaki azalma göz önüne alındığında, SSD'lerin toplam sahip olma maliyetlerinin çok daha düşük olduğu ve bilgi işlem merkezlerine her yıl milyonlarca lira tasarruf sağladığı görülmektedir.

2- NAND Flash Tabanlı SLC (Tek Katmanlı Hücre) veya MLC (Çok Katmanlı Hücre) SSD'ler güvenli değil

Gerçek: Genel olarak bakıldığında hem SLC hem de MLC NAND Flash tabanlı diskler sabit disklerden çok daha güvenilirdir. Çünkü mekanik parçaları yoktur. Hem SSD'ler hem de HDD'ler hata düzeltme kodu (ECC) kullanmaktadır. Her iki tür sürücü içinde hata düzeltme gereksinimi, yaklaşık 12.5 terabayt okumada birdir. Ayrıca, SSD'lerin titreşim ve elektro manyetik parazitlere olan dayanıklılığı HDD'lerle ölçülemeyecek derecede yüksektir. SSD'lerin 1990'ların ortalarından beri askeri ve kritik uygulamalarda kullanılmasının nedeni bu sağlamlık ve güvenilirlikleridir.

3- Hacker'lar Solid State Sürücüleri Sabit Sürücüler'den Daha Hızlı Hackler'ler

Gerçek: Hem HDD, hem SSD'ler hackerlar tarafından kırılabilir. Bilgisayarların korunma seviyesi güvenlik yazılımlarıyla ilgilidir. SSD'lerin HDD'lere göre teorik olarak 1000 kat daha hızlı bir şekilde veriyi getirebilmesi göz önüne alındığında bu durum gayet normal, ancak SSD'lerin sadece hacker'lar için tasarlandığını düşünmek ve bunu SSD'lerin bir zayıflığı olarak göstermek doğru değildir.

4- Özellikle çok veri yazıldığında SLC veya MLC Flash NAND tabanlı SSD'lerin ömrü kısadır

Gerçek: Flash SSD'lerdeki her bir hücrenin yazma dayanıklılık ömrü günümüzde 2 milyon döngüdür, yani tek bir hücre üzerine güvenli olarak 2 milyon defa yazılabilir. SSD'lerde bir bilginin yazılması için herhangi bir yazma kafasına veya döner bir bileşene ihtiyaç yoktur. Yazılacak hücre adreslenir ve doğrudan o adresteki hücre üzerine silme-yazma işlemi yapılır. Yazma ömrünü uzatmak için aynı hücre üzerine defalarca yazılmaz, bunun yerine sunucuyu ilgilendirmeyen bir algoritma kullanılarak bilgiler milyarlarca hücre arasından istatistikî olarak en az kullanılan hücreler üzerine yazılır. 64 GB'lık tek bir Flash SSD kullandığımızı (bir dizinde daha fazla ve yüksek kapasiteli sürücülerin kullanılması çalışma ömrünü uzatacaktır) ve yazma hızımızın 80Mbytes/s olduğunu varsayarsak (ki bu günümüzdeki en hızlı flash SSDlerin ortalama yazma hızıdır) 64 GB'lık bir SSD flash diskin her bir hücresinin üzerine minimum yazma-dayanıklılığı olan 2.000.000 defa silip yazmamız 55 yılımızı alacaktır. Buda sadece bilgi teknolojisi için değil, biz insanlar için bile oldukça uzun bir süredir.

5- SSD'lerin kapasiteleri çok düşük

Gerçek: Kurumsal-sınıf SSD aygıtı üreticisi olan Violin Memory Inc. 2U'luk bir şasi içinde optimize edilmiş RAID4 özelliklerine haiz 8 TB kullanılabilir kapasite sunabiliyor. 19" bir rack kabin içinde bunlardan 16 tanesi olduğu düşünülürse, tek bir kabinde 128 TB'lık bir kapasite elde edilebilmektedir. SSD'lerin kapasitesi sadece sunuculardaki yüklü olan işletim sisteminin adresleme kapasitesi ile sınırlıdır.

6- SSDlerin veri koruma ve kurtarma özellikleri çok zayıf

Gerçek: Kullanılan veri koruma yazılımlarının haricinde, ayrı disk dizinlerine ve RAID kontrolörlerine ihtiyaç duyan HDD ve diğer SSD sistemlerinin aksine, Violin1010'nun mimari yapısını oluşturan Violin Anahtarlanmış Bellek teknolojisi, maksimum veri koruma performans, yoğunluk ve enerji verimi için RAID Kontrolörünü bünyesinde bütünleşik olarak barındırmaktadır. 16KB'lik veri 4K'lık bloklar halinde ayrılmakta ve her bir 4K'lık veri farklı RAID grupları üzerindeki farklı modüllere 1K olarak yazılmakta, pariteleri ise RAID grubunu teşkil eden 5. modül üzerinde tutulmaktadır. Bu mimari ve RAID algoritması verinin en üst seviyede korunmasını sağlamaktadır. Violin 1010 cihazı içindeki 84 modülden herhangi ikisi arızalandığında bile, önce dört adet canlı yedek modüllerden biri otomatik olarak arızalanan modülün yerini almakta ve RAID algoritması ile korunan veri, önce canlı yedek modüllerden herhangi biri üzerine aktarılmakta, RAID grubu birkaç dakika içinde (HDD'lerden çok daha hızlı olarak) sistemdeki herhangi bir yedek evrensel modül üzerine otomatik olarak yeniden yapılanmakta ve yapılandırılmadan sonra, arızalı modüller sistem ve programlar çalışırken çıkartılarak yeni modüller ile değiştirilebilmekte (hot swapping) ve canlı yedek modül tekrar eski işlevine geri dönmektedir. Hem HDD'ler hem de SSD'ler işletim sistemi tarafından fiziksel bir depolama cihazı olarak görülmektedir. Bu nedenle bilgi işlem merkezinin kullandığı hata giderme yazılımına bağlı olarak, HDD'ler kullanıldığında saatlerce süren veri kurtarma işlemi SSD'ler

kullanıldığında dakikalar içinde tamamlanabilmektedir.

7- Bütün SSD'ler aynı performans ve güvenilirliğe sahip

Gerçek: Bütün SSD cihazları eşit olarak yaratılmamışlardır. Bir disk kontrol biriminin alınıp, bir NAND flash ile aynı kutuya koyulması başarılı bir SSD elde edilmesini sağlamaz. SSD mimarisinin bir bütün olarak ele alınması gerekir. Örneğin, kullanılan disk kontrol birimi, NAND teknolojisi ve genel SSD mimarisinin güvenilirlik, ölçeklilik, dayanıklılık, yıpranma dengeleme ve performans üzerinde önemli etkileri vardır. Ancak patenti Violin Memory Inc'e ait olan anahtarlanmış bellek mimarisi incelendiğinde, bu mimarinin neden çok yüksek sürdürebilir "350K read" ve "220K write" performansı gösterdiği, kullanıcılara donanıma yerleşik RAID4 özelliğinin yanı sıra, nasıl daha yüksek kullanılabilir kapasite sağlayabildiği açıkça görülebilir.

8- SSD'lerin sağladığı enerji tasarrufu o kadar önemli değil

Gerçek : Solid State Sürücülerde, kullanılan bellek modülleri için önemli olan parametre her GByte için ihtiyaç duyulan enerjidir. Bu nedenle enerji verimi, bilgi işlem merkezlerindeki sistemlerin toplam sahip olma maliyetini düşürmek için en önemli parametredir. Genel olarak, 3-5 yıllık ömürleri boyunca, donanımların enerji maliyetleri yatırım maliyetlerinin üstüne çıkmaktadır. Sadece dizüstü bilgisayarlar ve kişisel bilgisayarlar düşünülse bile, HDD'lerle kıyaslandığında "read" işlemleri için SSD'lerin harcadığı enerji yaklaşık %50, "write işlemleri için %30 daha azdır. Bir bilgi işlem merkezinde kullanılan disk dizinleri ve bunların bağlı olduğu sunucular düşünüldüğünde, gerek duyulan sunucu ve disk sayısındaki azalma nedeniyle elde edilen enerji tasarrufu oldukça önemlidir. Buna birde artık gerek duyulmayan bu sunucuların ve HDD'lerin soğutulması için harcanan enerji tüketimi eklendiğinde elde edilen tasarruf %80 seviyelerine çıkmaktadır.

9- HDD veriyi SSD'lerden daha uzun bir süre güvenli şekilde depolayabilir.

Gerçek: HDD'lerin write döngüsü açısından belirlenmiş herhangi bir dayanıklılık limiti olmamasına rağmen, bozulma oranları daha yüksek olup, aniden arızalanabilirler. SSD'lerin dayanıklılığının veya güvenilirliğinin kullanıcılar için bir endişe teşkil etmesi gerekmemektedir. Asıl endişe edilmesi gereken durumun bir HDD veya bir SSD dizini arızalandığında, verilerin ne kadar sürede geri yüklenebileceği ve buna bağlı olarak üretimde ne kadarlık bir kayba uğranacağı olmalıdır. HDD'lerin çalışma ömrü genelde 3-5 yıldır. Violin Memory Inc. Türkiye'de kendi SSD ürünleri için hem dayanıklılık hem de veri depolama özellikleri için en az 5 yıllık bir garanti vermektedir.

10- SSD'ler çok kısa bir süre içinde HDD'lerin yerini alacaktır

Gerçek: SSD'lerin performans, toplam sahip olma maliyeti ve güvenirlilik açısından çok büyük avantajları olmasına rağmen, yeni teknolojiye geçişte karşılaşılan bir çok gereksiz endişe, psikolojik ve ihtiyati nedenlerle daha uzun bir süre HDD'ler ile bir arada var olacağı beklenmektedir. Ancak IDC verilerine göre silikon depolama pazarı inanılmaz bir büyüme göstermektedir. Birkaç yıl içinde silikon depolamanın çok yaygın olarak kullanılacağı beklenmektedir.

Sonuç olarak yukarıda bahsi geçen noktalarda dikkate alındığında SSD disklerin Bilgi İşlem sistemleri için ne kadar önemli olduğu ve birçok kazanım sağladığı açıkça görülmektedir. SSD ile ilgili avantajları bir kez daha özetlemek gerekirse aşağıdaki maddeleri dile getirebiliriz.

- Çok daha yüksek IO performansı
- Hareketli parça olmamasından dolayı daha sessiz çalışma
- Çok daha az enerji tüketimi ve daha az ısınma
- Hareketli parça olmamasından dolayı daha fazla dayanıklılık
- Fiziksel olarak daha az alan kaplama
- Okuma ve yazma işlemlerinde daha az hata
- İşletme maliyetlerinin daha düşük olması
- Daha düşük Toplam Sahip Olma (TCO) maliyeti

File5

Same as File2 in Table 4.18.

File6

Same as File3 in Table 4.18.

File7

Same as File4 in Table 4.18.

File8
Same as File5 in Table 4.18.
File9
Same as File6 in Table 4.18.

Table 4.36. The Similarity Results of JPlag with m=2 token

JPlag	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	78	78	27	30	95	96	5	10
File2	40	x	99	20	22	40	41	5	5
File3	40	98	x	20	22	40	64	5	5
File4	6	8	8	x	99	5	7	2	2
File5	5	7	7	81	x	5	7	1	2
File6	96	78	78	27	28	x	96	5	7
File7	78	64	64	28	33	77	x	6	9
File8	17	34	35	29	32	16	23	x	97
File9	26	29	29	31	40	18	31	78	x

Table 4.37. The Similarity Results of Ferret with m=3 token

Ferret	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	33,02	32,93	0,18	0,85	89,25	71,74	0	2,73
File2	x	x	94,45	0,15	0,13	33,11	29,99	0,77	0,75
File3	x	x	x	0,15	0,13	33,02	29,91	0,77	0,75
File4	x	x	x	x	81,16	0,18	0,17	0,07	0,07
File5	x	x	x	x	x	0,15	0,63	0,11	0,77
File6	x	x	x	x	x	x	70,43	0	0
File7	x	x	x	x	x	x	x	0,2	2,13
File8	x	x	x	x	x	x	x	x	85,07
File9	x	x	x	x	x	x	x	x	x

Table 4.38. The Similarity Results of CopyCatch Gold with m=5 character

CopyCatch Gold	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	63	63	23	23	97	91	20	26
File2	x	x	99	30	31	63	61	23	24
File3	x	x	x	30	31	63	61	23	24
File4	x	x	x	x	93	23	27	16	18
File5	x	x	x	x	x	22	28	16	19
File6	x	x	x	x	x	x	89	20	22
File7	x	x	x	x	x	x	x	24	30
File8	x	x	x	x	x	x	x	x	92
File9	x	x	x	x	x	x	x	x	x

Table 4.39. The Similarity Results of WCopyfind with m=2 token

WCopyfind	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	77	77	6	10	94	94	2	7
File2	35	x	98	4	6	36	36	1	1
File3	35	98	x	5	6	36	36	1	1
File4	1	2	2	x	99	0	1	0	0
File5	1	2	2	82	x	0	1	0	1
File6	95	79	79	4	4	x	95	2	2
File7	76	63	63	5	8	76	x	2	5
File8	7	10	10	6	8	7	9	x	100
File9	17	9	9	7	18	6	16	87	x

Table 4.40. The Similarity Results of SIM with m=2 token

SIM	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	76	76	9	15	94	94	2	6
File2	36	x	98	10	14	39	38	1	1
File3	36	98	x	11	15	39	38	1	1
File4	2	5	5	x	99	4	4	1	1
File5	2	5	6	81	x	3	5	1	2
File6	94	83	83	17	21	x	95	2	2
File7	76	66	66	16	24	77	x	3	8
File8	6	8	8	17	22	6	10	x	96
File9	15	7	7	17	31	5	24	85	x

Table 4.41. The Similarity Results of Sherlock with m=2 token

Sherlock	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	33	34	0	0	89	69	0	0
File2	x	x	98	0	1	34	30	0	0
File3	x	x	x	0	1	35	30	0	0
File4	x	x	x	x	79	0	0	0	0
File5	x	x	x	x	x	0	0	0	1
File6	x	x	x	x	x	x	72	0	0
File7	x	x	x	x	x	x	x	2	2
File8	x	x	x	x	x	x	x	x	77
File9	x	x	x	x	x	x	x	x	x

Table 4.42. The Similarity Results of Text_Matching

Text_Matching	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	71	71	30	31	99	93	32	33
File2	x	x	99	41	43	70	73	42	43
File3	x	x	x	41	43	70	73	42	42
File4	x	x	x	x	97	27	40	34	38
File5	x	x	x	x	x	27	42	37	41
File6	x	x	x	x	x	x	90	30	30
File7	x	x	x	x	x	x	x	42	45
File8	x	x	x	x	x	x	x	x	96
File9	x	x	x	x	x	x	x	x	x

Table 4.43. The Similarity Results of Document Similarity–Cosine with m=2 token

Document Similarity - Cosine	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	57,34	57,95	5,74	6,83	96,03	88,85	4,77	10,41
File2	x	x	98,67	6,11	6,83	57,75	52,75	4,84	5,03
File3	x	x	x	6,44	7,38	57,75	54,06	4,82	6,13
File4	x	x	x	x	92,68	5,05	6,67	3,37	4,4
File5	x	x	x	x	x	5,05	8,73	38,98	7,04
File6	x	x	x	x	x	x	87,34	4,79	5,63
File7	x	x	x	x	x	x	x	7,08	15,16
File8	x	x	x	x	x	x	x	x	92,41
File9	x	x	x	x	x	x	x	x	x

Table 4.44. The Similarity Results of Document Similarity–Jaccard with m=2 token

Document Similarity - Jaccard	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	31,52	32,22	0,08	1,43	88,78	73,67	1,28	4,52
File2	x	x	96	1,45	1,68	32,02	30,11	1,6	1,7
File3	x	x	x	1,63	1,88	32,13	30,76	1,6	2,27
File4	x	x	x	x	81,8	0,77	1,08	0,63	0,86
File5	x	x	x	x	x	0,83	1,62	0,73	1,58
File6	x	x	x	x	x	x	74,24	1,27	1,95
File7	x	x	x	x	x	x	x	1,34	4,19
File8	x	x	x	x	x	x	x	x	84,73
File9	x	x	x	x	x	x	x	x	x

Table 4.45. The Similarity Results of Document Similarity–Dice with m=2 token

Document Similarity - Dice	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	47,94	48,74	1,66	2,82	94,06	84,85	2,53	8,65
File2	x	x	97,96	2,86	3,3	48,51	46,29	3,15	3,34
File3	x	x	x	3,2	3,69	48,63	47,05	3,14	4,44
File4	x	x	x	x	89,98	1,52	2,13	1,26	1,71
File5	x	x	x	x	x	1,65	3,19	1,44	3,1
File6	x	x	x	x	x	x	85,21	2,51	3,83
File7	x	x	x	x	x	x	x	2,65	8,03
File8	x	x	x	x	x	x	x	x	91,74
File9	x	x	x	x	x	x	x	x	x

Table 4.46. The Similarity Results of Document Similarity-Tanimoto with m=2 token

Document Similarity - Tanimoto	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	36,81	37,3	2,22	2,45	92,36	78,33	1,96	4,59
File2	x	x	97,37	2,88	3,07	37,09	34,78	1,54	1,71
File3	x	x	x	3,05	3,33	37,04	35,93	1,54	2,09
File4	x	x	x	x	85,34	1,94	2,84	0,75	1,05
File5	x	x	x	x	x	1,8	3,5	0,78	1,53
File6	x	x	x	x	x	x	75,91	1,98	2,44
File7	x	x	x	x	x	x	x	2,66	6,21
File8	x	x	x	x	x	x	x	x	85,4
File9	x	x	x	x	x	x	x	x	x

Table 4.47. The Results of our implemented Cosine Similarity

Cosine	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	73	72,74	50,89	52,25	97,94	93,7	17,76	21,76
File2	x	x	99,51	58,75	60,89	73,58	71,38	28,5	28,74
File3	x	x	x	58,78	60,92	73,31	71,24	29,44	29,74
File4	x	x	x	x	98,1	51,53	53,21	24,18	25,18
File5	x	x	x	x	x	52,59	54,99	25,13	26,68
File6	x	x	x	x	x	x	92,32	17,86	18,79
File7	x	x	x	x	x	x	x	23,67	28,92
File8	x	x	x	x	x	x	x	x	94,08
File9	x	x	x	x	x	x	x	x	x

Table 4.48. The Results of our implemented Dice Similarity

Dice	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	63,34	62,93	26,72	24,17	97,94	92,74	14,35	18,36
File2	x	x	99,51	46,4	43,71	64,14	65,98	15,45	16,6
File3	x	x	x	46,58	43,9	63,73	65,71	15,88	17,1
File4	x	x	x	x	97,07	27,28	31,48	6,83	7,66
File5	x	x	x	x	x	24,54	28,83	6,17	7,06
File6	x	x	x	x	x	x	91,5	14,35	15,77
File7	x	x	x	x	x	x	x	17,45	22,43
File8	x	x	x	x	x	x	x	x	93,8
File9	x	x	x	x	x	x	x	x	x

Table 4.49. The Results of our implemented Jaccard Similarity

Jaccard	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	46,35	45,91	15,42	13,75	95,96	86,46	7,73	10,11
File2	x	x	99,03	30,2	27,97	47,21	49,23	8,37	9,05
File3	x	x	x	30,36	28,13	46,76	48,93	8,63	9,35
File4	x	x	x	x	94,31	15,79	18,68	3,54	3,98
File5	x	x	x	x	x	13,99	16,84	3,18	3,66
File6	x	x	x	x	x	x	84,33	7,73	8,56
File7	x	x	x	x	x	x	x	9,56	12,63
File8	x	x	x	x	x	x	x	x	88,32
File9	x	x	x	x	x	x	x	x	x

Table 4.50. The Similarity Results of DOCCop with m=8 token

DOCCop	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	58	58	0	3	96	87	0	8
File2	x	x	99	0	0	58	51	0	0
File3	x	x	x	0	0	58	51	0	0
File4	x	x	x	x	91	0	0	0	0
File5	x	x	x	x	x	0	3	0	6
File6	x	x	x	x	x	x	87	0	0
File7	x	x	x	x	x	x	x	0	8
File8	x	x	x	x	x	x	x	x	91
File9	x	x	x	x	x	x	x	x	x

Table 4.51. The Results of Turnitin

Turnitin	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	0	0	0	5	95	96	0	0
File2	0	x	99	0	0	0	0	0	0
File3	37	99	x	0	0	0	37	0	0
File4	0	0	0	x	100	0	0	0	0
File5	1	0	0	81	x	0	0	0	1
File6	96	0	0	0	0	x	96	0	0
File7	79	0	0	0	0	77	x	0	0
File8	0	0	0	0	0	0	0	x	97
File9	13	0	0	0	13	0	13	84	x

Table 4.52. The Proposed Algorithm Results As a Table for Scenario 28 with m=1

GST (m=1)	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	88,13	88,49	48,92	54,32	96,04	97,84	17,63	21,22
File2	44,71	x	99,45	45,62	50,73	43,98	48,18	12,59	13,69
File3	44,73	99,09	x	46	51,09	44	48,18	12,73	13,82
File4	11,87	21,82	22,08	x	100	11,87	15,18	5,85	6,63
File5	10,58	19,48	19,69	80,31	x	10,3	13,88	5,05	5,96
File6	97,09	87,64	88	49,45	53,45	x	96,73	17,82	19,27
File7	79,3	76,97	77,26	50,73	57,73	77,55	x	15,74	19,24
File8	48,04	67,65	68,63	65,69	70,59	48,04	52,94	x	100
File9	48,36	61,48	62,3	62,3	69,67	43,44	54,1	83,61	x

Table 4.53. The Proposed Algorithm Results As a Table for Scenario 28 with m=2

GST (m=2)	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	77,7	79,5	12,23	16,19	94,96	96,4	5,76	10,43
File2	39,42	x	99,09	9,67	11,86	39,42	40,51	5,84	6,2
File3	40,18	98,73	x	10,36	12,55	39,64	40,91	5,82	6,73
File4	2,97	4,62	4,97	x	100	2,97	4,19	2,01	2,53
File5	3,15	4,56	4,84	80,31	x	2,66	4,63	1,68	2,59
File6	96	78,55	79,27	12,36	13,82	x	95,64	5,82	7,27
File7	78,13	64,72	65,6	13,99	19,24	76,68	x	5,25	9,04
File8	15,69	31,37	31,37	22,55	23,53	15,69	17,65	x	100
File9	23,77	27,87	30,33	23,77	30,33	16,39	25,41	83,61	x

Table 4.54. The Proposed Algorithm Results of Stopwords as a Table with m=1

Stopword	File1	File2	File3	File4	File5	File6	File7	File8	File9
File1	x	56,22	55,96	29,35	26,92	99,21	91,97	31,82	33,71
File2	x	x	99,68	55,47	50,49	58,72	62,28	23,46	23,33
File3	x	x	x	53,91	50	58,45	61,14	23,33	23,2
File4	x	x	x	x	92,71	30,35	34,83	11,52	12,08
File5	x	x	x	x	x	28,3	32,79	10,58	11,08
File6	x	x	x	x	x	x	92,75	31,46	33,33
File7	x	x	x	x	x	x	x	30,3	32
File8	x	x	x	x	x	x	x	x	98,04
File9	x	x	x	x	x	x	x	x	x

Table 4.55. The Proposed Algorithm Results of Punctuations as a Table with m=1

Punctuation	**File1**	**File2**	**File3**	**File4**	**File5**	**File6**	**File7**	**File8**	**File9**
File1	x	46,38	47,76	29,7	23,61	94,12	70,83	64,71	71,79
File2	x	x	98,04	72,06	56,98	49,28	67,47	49,28	62,16
File3	x	x	x	70,15	55,37	50,75	69,14	50,75	63,89
File4	x	x	x	x	79,62	31,68	46,96	33,66	37,74
File5	x	x	x	x	x	25	40,51	25	29,53
File6	x	x	x	x	x	x	70,83	64,71	76,92
File7	x	x	x	x	x	x	x	58,33	71,7
File8	x	x	x	x	x	x	x	x	76,92
File9	x	x	x	x	x	x	x	x	x

Table 4.57. The Comparison of the Running Time of All Similarity Tools

ASSIGNMENT	SOFTWARE	MINIMUM MATCH LENGTH	TIME (sc)
1_1	WCopyfind	m=1	59
1_1	WCopyfind	m=2	53
1_1	GST	m=1	293
1_1	GST	m=2	275
1_1	SIM	m=1	3
1_1	SIM	m=2	1
1_1	Sherlock	m=1	1
1_1	Sherlock	m=2	1
1_1	Text_Matching	-	23
1_1	CopyCatch Gold	m=5	2
1_1	JPlag	m=2	60
1_1	Our implemented Cosine, Dice, Jaccard	-	4
1_1	Turnitin	-	248
1_2	WCopyfind	m=1	59
1_2	WCopyfind	m=2	52
1_2	GST	m=1	439
1_2	GST	m=2	355
1_2	SIM	m=1	2
1_2	SIM	m=2	1
1_2	Sherlock	m=1	1
1_2	Sherlock	m=2	1
1_2	Text_Matching	-	12
1_2	CopyCatch Gold	m=5	2
1_2	JPlag	m=2	68
1_2	Our implemented Cosine, Dice, Jaccard	-	6
1_2	Turnitin	-	484
1_3	WCopyfind	m=1	42
1_3	WCopyfind	m=2	50
1_3	GST	m=1	331
1_3	GST	m=2	319
1_3	SIM	m=1	1
1_3	SIM	m=2	1
1_3	Sherlock	m=1	1
1_3	Sherlock	m=2	1
1_3	Text_Matching	-	12
1_3	CopyCatch Gold	m=5	1
1_3	JPlag	m=2	62
1_3	Our implemented Cosine, Dice, Jaccard	-	5
1_3	Turnitin	-	188
2_1	WCopyfind	m=1	42
2_1	WCopyfind	m=2	65
2_1	GST	m=1	280
2_1	GST	m=2	242
2_1	SIM	m=1	3
2_1	SIM	m=2	1
2_1	Sherlock	m=1	1
2_1	Sherlock	m=2	1

Table 4.57. (Continued)

ASSIGNMENT	SOFTWARE	MINIMUM MATCH LENGTH	TIME (sc)
2_1	Text_Matching	-	15
2_1	CopyCatch Gold	m=5	1
2_1	JPlag	m=2	66
2_1	Our implemented Cosine, Dice, Jaccard	-	4
2_1	Turnitin	-	65
2_2	WCopyfind	m=1	40
2_2	WCopyfind	m=2	33
2_2	GST	m=1	310
2_2	GST	m=2	270
2_2	SIM	m=1	1
2_2	SIM	m=2	1
2_2	Sherlock	m=1	1
2_2	Sherlock	m=2	1
2_2	Text_Matching	-	8
2_2	CopyCatch Gold	m=5	2
2_2	JPlag	m=2	58
2_2	Our implemented Cosine, Dice, Jaccard	-	4
2_2	Turnitin	-	186
2_3	WCopyfind	m=1	63
2_3	WCopyfind	m=2	49
2_3	GST	m=1	1079
2_3	GST	m=2	909
2_3	SIM	m=1	2
2_3	SIM	m=2	1
2_3	Sherlock	m=1	1
2_3	Sherlock	m=2	1
2_3	Text_Matching	-	10
2_3	CopyCatch Gold	m=5	1
2_3	JPlag	m=2	83
2_3	Our implemented Cosine, Dice, Jaccard	-	5
2_3	Turnitin	-	427
3	WCopyfind	m=1	56
3	WCopyfind	m=2	53
3	GST	m=1	286
3	GST	m=2	267
3	SIM	m=1	2
3	SIM	m=2	1
3	Sherlock	m=1	1
3	Sherlock	m=2	1
3	Text_Matching	-	14
3	CopyCatch Gold	m=5	1
3	JPlag	m=2	59
3	Our implemented Cosine, Dice, Jaccard	-	3
3	Turnitin	-	300

Table 4.57. (Continued)

ASSIGNMENT	SOFTWARE	MINIMUM MATCH LENGTH	TIME (sc)
4	WCopyfind	m=1	43
4	WCopyfind	m=2	44
4	GST	m=1	667
4	GST	m=2	630
4	SIM	m=1	2
4	SIM	m=2	1
4	Sherlock	m=1	1
4	Sherlock	m=2	1
4	Text_Matching	-	14
4	CopyCatch Gold	m=5	1
4	JPlag	m=2	55
4	Our implemented Cosine, Dice, Jaccard	-	4
4	Turnitin	-	175
5	WCopyfind	m=1	14
5	WCopyfind	m=2	14
5	GST	m=1	139
5	GST	m=2	135
5	SIM	m=1	1
5	SIM	m=2	1
5	Sherlock	m=1	1
5	Sherlock	m=2	1
5	Text_Matching	-	7
5	CopyCatch Gold	m=5	2
5	JPlag	m=2	19
5	Our implemented Cosine, Dice, Jaccard	-	3
5	Turnitin	-	118

Table 4.60. The F-Measure Comparison of All Classification Algorithms for the raw dataset with all attributes

CLASSIFIER	TYPE
ADTree	Tree-Based Classifier
AODE	Bayesian Network Classifier
AODEsr	Bayesian Network Classifier
BayesNet	Bayesian Network Classifier
BFTree	Tree-Based Classifier
ConjunctiveRule	Rule-Based Classifier
DecisionStump	Tree-Based Classifier
DecisionTable	Rule-Based Classifier
DTNB	Rule-Based Classifier
FT	Tree-Based Classifier
HNB	Bayesian Network Classifier
HyperPipes	Miscellaneous Classifier
IB1	Lazy Classifer (Instance-Based Classifier)
IBk (k=3)	Lazy Classifer (Instance-Based Classifier)
Id3	Tree-Based Classifier
J48	Tree-Based Classifier
J48graft	Tree-Based Classifier
JRip	Rule-Based Classifier
KStar	Lazy Classifer (Instance-Based Classifier)
LADTree	Tree-Based Classifier
LBR	Lazy Classifer (Instance-Based Classifier)
LMT	Tree-Based Classifier
Logistic	Function-Based Classifier
LWL	Lazy Classifer (Instance-Based Classifier)
MultilayerPerceptron	Function-Based Classifier
NaiveBayes	Bayesian Network Classifier
NaiveBayesSimple	Bayesian Network Classifier
NaiveBayesUpdateable	Bayesian Network Classifier
NBTree	Tree-Based Classifier
NNge	Rule-Based Classifier
OneR	Rule-Based Classifier
PART	Rule-Based Classifier
Prism	Rule-Based Classifier
RandomForest	Tree-Based Classifier
RandomTree	Tree-Based Classifier
RBFNetwork	Function-Based Classifier
REPTree	Tree-Based Classifier
Ridor	Rule-Based Classifier
SimpleCart	Tree-Based Classifier
SimpleLogistic	Function-Based Classifier
SMO	Function-Based Classifier
SPegasos	Function-Based Classifier
VFI	Miscellaneous Classifier
VotedPerceptron	Function-Based Classifier
WAODE	Bayesian Network Classifier
Winnow	Function-Based Classifier
ZeroR	Rule-Based Classifier

Table 4.65. Comparison of Our Proposal With the Related Studies The Comparison of Literature Review

Reference	CMS / LMS used	Language	Plagiarism Detection Tool / Algorithms	Plagiarism Type	E-Assessment Type
Gutiérrez et al. [13]	Moodle	VHDL	No (by using team identifier of the student groups)	Source Code	Automatic
Ramos et al. [14]	Moodle	Matlab, VHDL	No (But plagiarism may be restricted with team id)	Source Code	Automatic
Spinellis et al. [15]	No	Java	No (by generating randomized programming assignments)	Source Code	Automatic
Rodriguez et al. [16]	No	Assembly	Yes (a plagiarism detection assistant)	Source Code	Automatic
Tresnawati et. al.	Moodle		Yes (SMS Gateway)	Source Code	No Info
Amelung et al. [17]	Plone	Phyton, Haskell, Scheme, Common Lisp and Prolog	No	Source Code	Automatic and Semi-automatic
Malmi et al. [19]	TRAKLA and Ceilidh	C, Fortran, Scheme, and Java	No (by personalizing the assignments)	Source Code	Automatic
Liu [22]	No	C and Delphi	No	Source Code	Automatic
Gutiérrez et al. [23]	Moodle	VHDL/Matlab	No (But plagiarism may be restricted with team name)	Source Code	Automatic
Marques et al. [24]	Virtual Campus	No Info (used for evaluating algorithms for distributed systems)	JPlag	Source Code	Automatic
Bejarano et al. [25]	No	C++	Yes	Source Code	None
Gao [26]	No	Html	Fingerprinting	Source Code	None
Simoes et al. [27]	PLEBOX	None	No	Source Code	None
Ramirez et al. [28]	Athena	No Info	No	Source Code	No Info
Gitchell and Tran	No	C	Sim (using string alignment techniques)	Source Code	None
Riad et. al. (2013)	Yes	No Info	ZPlag (using tokenization technique for source code, similarity algorithm for textual documents, and vector features representation for images)	Source Code, Textual, Images	No Info
Chen *et. al*	No	English	*n*-gram analysis up to 4-gram, LCS, skip-bigram, and WordNet	Textual	No Info
Jayapal (2012)	No	English	*N*-gram similarity overlap metric, The Greedy String Tiling Algorithm	Textual	No Info

Table 4.65. (Continued)

Reference	CMS / LMS used	Language	Plagiarism Detection Tool / Algorithms	Plagiarism Type	E-Assessment Type
Fredriksson and Navarro (2004)	No	English	Yes (ℓ-gram)	Textual	None
Su *et. al.*	No	English	Yes (Levenshtein Distance and Smith-Waterman Algorithm)	Textual	None
Wise	No	English	Yes, Running Karp-Rabin Greedy String Tiling (RKR-GST) Algorithm	Textual	None
Işık and Çamurcu	No	Turkish	Euclidean, Cosine, Pearson and Extended Jaccard similarity measures	Textual	None
Yücesoy (2007)	No	Turkish	Lexical semantic similarity method, Wu-Palmer method	Textual	None
Dursun	No	Turkish	The improved similarity computing method, Edit Distance Similarity, and Jaro-Winkler Similarity	Textual	No Info
Alzahrani *et. al.* (2009)	Yes	Arabic	Arabic Plagiarism Detection (APD) (using Google API and 4-grams)	Textual	No Info
Menai and Bagais (2011)	No Info	Arabic	APlag (The Longest Common Substring algorithm)	Textual	No Info
Menai (2012)	No Info	Arabic	APlag (The Longest Common Substring algorithm)	Textual	No Info
Gruner and Naven	No	English	Yes (a stylometry based tool)	Textual	No Info
Shcherbinin and Butakov (2008)	Moodle	No Info	Winnowing Algorithm, MSN LiveTM search engine	Textual, Source Code	Manual
Butakov and Shcherbinin (2009)	Moodle	Language independent	Modified Winnowing Algorithm, Microsoft Live Search service	Textual	None
Shcherbinin and Butakov (2009)	Moodle	Language independent	Winnowing Algorithm, Levenstein Distance, Microsoft SQL Server	Textual, Source Code	None
Jiffriya *et. al.* (2013)	No	English	Tri-gram sequence matching technique, K-Means	Textual	None

Table 4.65. (Continued)

Reference	CMS / LMS used	Language	Plagiarism Detection Tool / Algorithms	Plagiarism Type	E-Assessment Type
Source-Code Plagiarism Plugin [34]	Moodle	Java, C#, C, C++, Scheme, Python, Visual Basic, Javascript, etc.	Yes, Moss and JPlag	Source Code	None
Moss Plugin [35]	Moodle	C, C++, Java, C#, Python, Visual Basic, Javascript, etc.	Yes, Moss	Source Code	None
C_Programming [36]	Moodle	C	No	Source Code	None
Virtual Programming Lab (VPL) [37]	Moodle	Ada, C, C++, C#, Fortran, Haskell, Java, Matlab/Octave, Pascal, etc.	Yes, its own plagiarism detection algorithm	Source Code	Automatic and Semi-automatic
Kaya and Özel (2014)	Moodle	C (and all languages supported by GCC and Moss)	Yes (Moss)	Source Code	Semi-automatic
Our Proposal	Moodle	Turkish	Yes, Greedy String Tiling Algorithm and Cosine Similarity	Textual	Semi-automatic

Table 4.66. Comparison of study with the related studies about the web usage mining

Reference	Attribute Number	Student Number	LMS	Tool	Discretization Method	Data Mining Method	Plagiarism	Grade	Student Group Number	F-Measure	10-fold cross validaiton / train-test data
El-Halees (2009)	9	151	Yes, Moodle	Knim	Yes	Classificaiton - J48 Decision Tree; Clustering - EM Clustering; Association Rule Mining - Apriori (Lift Metric)	No	Yes	5	NA	NA
Romero et. Al. (2010a)	9	138, 135, 438	Yes, Moodle	Keel	Yes - Equal Width	Classificaiton - several classifciation algorithms; Association Rule Mining - AprioriC	No	Yes	NA	%65 accuracy	10-fold cross validatio n
Dimic et. Al. (2010)	5	130	Yes, Moodle	Weka	Yes - Equal Width, manual method	Classificaiton - J48 Decision Tree; Clustering - EM Clustering; Association Rule Mining	No	Yes	3 intervals, 5 groups	NA	NA
Casey and Gibson (2010)	NA	250, 50, 89	Yes, Moodle	NA	NA	Association Rule Mining	No	Yes	NA	NA	NA
Pedraza-Perez et. Al. (2011)	NA	NA	Yes, Moodle	Keel	Yes	Classification, Association Rule Mining, Regression	No	Yes	NA	NA	Train-Test Data
Jailia and Tyagi (2013)	13	NA	Yes, Moodle	Rapid Miner	Yes	Classification - Decision Tree Algorithm	No	Yes	NA	NA	NA

Table 4.66. (Continued)

Reference	Attribute Number	Student Number	LMS	Tool	Discretization Method	Data Mining Method	Plagiarism	Grade	Student Group Number	F-Measure	10-Fold Cross validaiton / Train-Test Data
Mohd (2013)	5	30	Yes, Moodle	NA	NA	Clustering, Association Rule Mining	No	Yes	NA	NA	NA
Bovo *et. al.* (2013)	NA	NA	Yes, Moodle	Weka	NA	Clustering – several algorithms	No	Yes	2-3	NA	10-fold cross validaiton
Romero *et. al.* (2007)	12	438	Yes, Moodle	Weka, Keel	Yes, Equal Width, Manual Method	Clustering – K-Means Algorithm, Classification - C4.5 Algorithm; Association Rule Mining – Apriori Algorithm	No	Yes	3	NA	3-folds cross validation
Romero *et. al.* (2009)	11	293	Yes, Moodle	Keel	NA	Apriori-SD, CN2-SD, and CBA, SDIGA Algorithms	No	Yes	NA	NA	NA
Romero *et. al.* (2010b)	11	230	Yes, Moodle	NA	NA	Several Association Rule Mining (ARM) and RARM algorithms – Apriori-Frequent, Apriori-Infrequent, Apriori-Inverse, and Apriori-Rare	No	Yes	NA	NA	NA

Table 4.66. (Continued)

Reference	Attribute Number	Student Number	LMS	Tool	Discretization Method	Data Mining Method	Plagiarism	Grade	Student Group Number	F-Measure	10-Fold Cross validaiton / Train-Test Data
García *et. al.* (2010)	12		Yes, Moodle	Weka	Yes, Equal Width, Manual Method	Association Rule Mining – Apriori Algorithm	No	Yes	NA	NA	NA
Lile (2011)	11	29	Yes, Moodle	Rapid Miner, Weka	NA	Attribute Selection – Information Gain, Relief, Chi-Squared, and Uncertainty; Clustering – K-Means; Classification – Tree Induction; Association Rule Mining – Apriori, FPGrowth, and Generalized Sequential Patterns (GSP)	No	Yes	NA	NA	NA
Kularbphettong and Tongsiri (2012)	19	3140 records	Yes, Moodle	Weka	Yes, Equal Width	Association Rule Mining – Apriori Algorithm, Classification Algorithms	No	Yes	NA	Between 68.53% and 83.92% accuracy	Train and Test Data

Table 4.66. (Continued)

Reference	Attribute Number	Student Number	LMS	Tool	Discretization Method	Data Mining Method	Plagiarism	Grade	Student Group Number	F-Measure	10-Fold Cross validaiton / Train-Test Data
Chajri *et. al.* (2014)	11	30	Yes, Moodle	Rapid Miner	Yes, Equal Width	Attribute Selection – Information Gain; Classification – Tree Decision; Clustering – K-Means; and Association Rule Mining – FP-Growth	No	Yes	NA	85.67% accuracy	NA
Our Proposal	31	44 students, 155 records	Yes, Moodle	Weka and Keel	Yes, Equal Frequency	Classification – All Algorithms; and Association Rule Mining – Apriori Algorithm	Yes	Yes	For Plagiarism = 2; For Grade = 3	For Plagiarism 0.702; For Grade 0.994	10-fold cross validation