

**ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES**

PhD THESIS

Furkan GÖZÜKARA

**PRODUCT SEARCH ENGINE USING PRODUCT NAME
RECOGNITION AND SENTIMENT ANALYSIS**

DEPARTMENT OF COMPUTER ENGINEERING

ADANA-2016

**ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES**

**PRODUCT SEARCH ENGINE USING PRODUCT NAME
RECOGNITION AND SENTIMENT ANALYSIS**

Furkan GÖZÜKARA

PhD THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the Master of Science by the board of jury on 19/10/2016.

.....
Assoc. Prof. Dr. Selma Ayşe ÖZEL
SUPERVISOR

.....
Assoc. Prof. Dr. Mutlu AVCI
MEMBER

.....
Assoc. Prof. Dr. Zekeriya TÜFEKÇİ
MEMBER

.....
Assoc. Prof. Dr. Zeki YETGİN
MEMBER

.....
Asst. Prof. Dr. Ali İNAN
MEMBER

This MSc Thesis is written at the Department of Institute of Natural And Applied Sciences of Çukurova University.

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

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) scholarship 2211-C.

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

ABSTRACT

PhD THESIS

PRODUCT SEARCH ENGINE USING PRODUCT NAME RECOGNITION AND SENTIMENT ANALYSIS

Furkan GÖZÜKARA

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES DEPARTMENT
OF COMPUTER ENGINEERING

Supervisor : Assoc. Prof. Dr. Selma Ayşe ÖZEL
Year: 2016, Pages: 212
Jury : Assoc. Prof. Dr. Selma Ayşe ÖZEL
: Assoc. Prof. Dr. Mutlu AVCI
: Assoc. Prof. Dr. Zekeriya TÜFEKÇİ
: Assoc. Prof. Dr. Zeki YETGİN
: Asst. Prof. Dr. Ali İNAN

In this study, a novel product search engine system which consists of a focused crawler, a record linkage system and a sentiment analyzer is proposed. We develop an original focused web crawler for E-commerce sites, and the challenges and our proposed solutions are presented in detail. A sentiment analyzer is developed to classify E-commerce product comments into polarities as negative or positive. A novel record linkage system for E-commerce products is proposed to recognize the same product names collected from different E-commerce sites. The record linkage system is based on a modified dynamic/incremental Hierarchical Agglomerative Clustering algorithm which employs our proposed product code matching system to reduce number of product name comparisons during clustering. In addition to these systems, a search system and a user interface are developed for the product search engine. In this thesis, we present a full scale product search engine that obtains %472 performance boosts in the crawler, 91.08% accuracy in the sentiment analysis, 96.25% F-measure in the record linkage, and 100% precision in most related products search. The proposed system achieves to provide better user experience than the existing systems.

Key Words: Sentiment Analysis, Data Mining, Record Linkage, Product Clustering, Focused Web Crawler

ÖZ

DOKTORA TEZİ

**DUYGU ANALİZİ VE ÜRÜN İSİM TANIMLAMA KULLANAN ÜRÜN
ARAMA MOTORU**

Furkan GÖZÜKARA

**ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

Danışman : Doç. Dr. Selma Ayşe ÖZEL
Yıl: 2016, Sayfa: 212
Jüri : Doç. Dr. Selma Ayşe ÖZEL
: Doç. Dr. Mutlu AVCI
: Doç. Dr. Zekeriya TÜFEKÇİ
: Doç. Dr. Zeki YETGİN
: Yrd. Doç. Dr. Ali İNAN

Bu çalışmada, odaklı tarayıcı, kayıt eşleştirme ve duygu analizi sistemlerinden oluşan özgün bir ürün arama motoru önerilmiştir. E-ticaret siteleri için özgün bir odaklı tarayıcı geliştirilmiş ve bu süreçteki zorluklar ve bu zorluklara önerilen çözüm önerileri detaylı bir şekilde sunulmuştur. E-ticaret ürünlerine yapılan yorumları olumlu ve olumsuz şeklinde sınıflandıracak bir duygu analizi sistemi geliştirilmiştir. Farklı E-ticaret sitelerinden toplanmış aynı ürün isimlerini tanıyacak özgün bir kayıt eşleştirme sistemi sunulmuştur. Kayıt eşleştirme sistemi, dinamik/artımsal olacak şekilde değiştirilmiş Hiyerarşik Tümevarım Kümeleme algoritması üzerine kurulmuştur ve önerdiğimiz ürün kodu eşleştirme sistemi ile kümeleme yaparken ürün ismi karşılaştırma sayısını oldukça azaltmaktadır. Bu sistemlere ek olarak, ürün arama motoru için bir arama sistemi ve kullanıcı arayüzü de geliştirilmiştir. Bu tez çalışmasında, web taramasında %472 performans artışı, duygu analizinde %91.08 doğruluk, kayıt eşleştirme sisteminde %96.25 F-ölçütü, ve en alakalı ürünleri arama sisteminde %100 isabet elde eden tam kapsamlı bir ürün arama motoru sunulmuştur. Önerilen sistem mevcut sistemlerden daha iyi bir kullanıcı deneyimi sunmayı başarmaktadır.

Anahtar Kelimeler: Duygu Analizi, Veri Madenciliği, Kayıt Eşleştirme, Ürün Kümeleme, Odaklı Web Tarayıcı

EXPANDED ABSTRACT

Due to the advances in E-commerce technologies and numerous available E-commerce websites, it is much harder for an average consumer to decide where to make an online purchase nowadays. Consumers are generally confused when going to make an online purchase since there are so many E-commerce websites, and the prices of products are constantly updated. To solve this problem, in this thesis, we have proposed an advanced product search engine which collects product information from various different E-commerce sites and uses product name recognition and sentiment analysis to provide consumers making search for “the most related products” and “the best products for a given category”. The proposed product search engine is composed of the following four main components:

- A focused web crawler to crawl and process E-commerce websites;
- A sentiment analysis system to classify E-commerce product’s comments into their contextual polarities as negative and positive;
- A clustering system that utilizes product code matching to recognize the same product names which are written differently and crawled from different E-commerce sites;
- A search system and user interface to allow users to make queries for the product that they request.

For developing the focused web crawler, C# programming language, .NET 4.5 framework and MS-SQL 2014 database management system are used. Moreover, HtmlAgilityPack and Newtonsoft.Json open source libraries are utilized. The challenges encountered during the development of the focused web crawler, and our proposed solutions are presented along with the experimental evaluations.

With the proposed system and possible solutions, up to 472% performance improvement with respect to baseline crawling strategies is achieved.

In our sentiment analyzer, Support Vector Machines are used as the classifier. The effects of applying preprocessing steps such as spelling correction, diacritics transformation, lower case conversion, data transformation, and punctuation removal over the performance of the sentiment analysis of user comments are investigated. For tokenization and vectorization, many state-of-the-art algorithms are experimented such as: term frequency, inverse document frequency, and relevance frequency weighting schemes with their variants. The effects of the most widely used feature selection methods that are information gain and chi-square over the sentiment analysis process are also experimented. For the Support Vector Machines classifier, best kernel type and parameter values are determined experimentally. In this study, a new dataset is composed of the product comments that are crawled by the focused web crawler for performance evaluations of the sentiment analyzer. This dataset is labeled by a domain expert manually into contextual polarities as negative and positive. By utilizing only statistical methods, without being depended on the language of the comments, our proposed system achieves 91.08% accuracy on our composed dataset and surpasses other experimented state-of-the-art algorithms. Moreover, the proposed preprocessing, tokenization and vectorization methods have better performance over the publicly available Turkish and English sentiment analysis datasets with respect to most of the previously proposed state-of-the-art methods.

The core part of the product search engine is clustering the same products that are given different names and sold by different E-commerce websites. This problem is an example of record linkage problem and to solve it, a clustering system that utilizes product code matching is proposed. The proposed record linkage system uses modified to be incremental Hierarchical Agglomerative Clustering algorithm along with the inclusion of our proposed matching functions that reduce number of product name comparisons. During the development of the

clustering system, many different state-of-the-art methods and algorithms are experimented such as: term frequency and inverse document frequency weighting with their variances, vector normalizations, N-gram transformations, distance metrics, linkage criteria, etc. For experimenting, an E-commerce products dataset is formed by collecting product information from 50 different E-commerce websites. Then, the collected data is labeled by a domain expert, and finally several datasets are formed from the labeled dataset by taking random subsets. With the help of the matching functions which utilizes price feature matching, supervised brand filtering, word conflict check, and product code matching, the proposed record linkage system achieves 96.25% F-measure while other experimented state-of-the-art algorithms achieve 89.12% F-measure at best. Moreover, we publish our datasets that are used in the experiments in both preprocessed and raw formats. To our knowledge, our datasets are the first publicly available E-commerce products datasets.

In this thesis, another problem that we try to solve is the ambiguity in the evaluation methods of the clustering tasks. In the literature, performance evaluation criterion of clustering is not precise, and there are different implementations for one of the most widely used performance metric which is F-measure. In this study, we propose to use ELKI, which is an open source framework, for performance evaluation of clustering tasks. Therefore, future researchers can use the same system and can make objective comparisons.

For users to interact and use the search system, a search engine, and user interface are developed. The developed search system supports two different kinds of queries. The first supported query type is “finding most related products” according to the query. Finding most related products feature helps users to choose which product to buy. The second supported query type is “listing best products” in a given category. The user interface shows a list of products and for each product it also displays which E-commerce websites sell the product from which prices. Moreover, users can easily display the cumulative comments made for products.

By utilizing the proposed sentiment analysis system, the comments are classified into negative and positive classes. Thus, users can easily read both positive and negative opinions about the products before deciding their purchase. The proposed search system is compared with the existing commercial product search engines and public search engines. Both the proposed system and some of the commercial product search engines achieve 100% precision for finding the most related products. However, best products in a given category are better supported by our system when compared to general search engines, and commercial search systems do not support this query type.

Conclusively, we believe that, the proposed solutions and used methods for developing our focused web crawler, sentiment analyzer, and record linkage method for E-commerce products will be invaluable for the future researchers and developers. The methods used by our focused crawler are also easily applicable to general crawlers for performance improvements. Our sentiment analysis system is language independent and does not use expensive resources such as dictionaries, therefore it can easily be applied to any language with high effectiveness. Our proposed clustering system achieves the best performance results in record linkage problem of E-commerce products, and it is very robust against overtraining according to our empirical analyses. Moreover, we introduce sentiment analysis and record linkage datasets to the literature along with systematic performance evaluations. Thus, it will be possible for future researchers to make objective comparisons.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my heart-whole thankfulness to my advisor Assoc. Prof. Dr. Selma Ayşe ÖZEL for her leading, patience, and active support during my PhD education.

I would like to thank Assoc. Prof. Dr. Mutlu AVCI and Assoc. Prof. Dr. Mustafa ORAL for being in my thesis committee and supporting me with their invaluable comments and contributions.

I would like to thank Assoc. Prof. Dr. Zeki YETGİN, Assoc. Prof. Dr. Zekeriya TÜFEKÇİ and Asst. Prof. Dr. Ali İNAN for honoring me by accepting to be jury members of my PhD thesis defense.

I would like to thank Scientific and Technological Research Council of Turkey (TÜBİTAK) for supporting this thesis by 2211-C scholarship.

Lastly, I would like to thank my parents and my wife for believing in me, supporting me and encouraging me throughout my life. Without their support, this thesis would not have been possible.

Furkan GÖZÜKARA

CONTENTS	PAGE
ABSTRACT	I
ÖZ.....	II
EXPANDED ABSTRACT.....	III
ACKNOWLEDGEMENTS	VII
CONTENTS	VIII
LIST OF TABLES	XIV
LIST OF FIGURES.....	XVI
LIST OF CODE SNIPPETS.....	XVIII
LIST OF ABBREVIATIONS	XX
1. INTRODUCTION.....	1
1.1. Preliminaries.....	2
1.1.1. CSA Systems	2
1.1.2. Wrappers.....	3
1.1.3. Focused Web Crawlers.....	3
1.1.4. Sentiment Analysis.....	4
1.1.5. Record Linkage	5
1.2. Summary of the Contributions	5
1.2.1. Contributions in the Focused Web Crawler Area.....	5
1.2.2. Contributions in the Sentiment Analysis Area	6
1.2.3. Contributions in the Record Linkage Area.....	7
1.2.4. Contributions in the Product Search Engine Area.....	8
1.3. Organization of the Thesis.....	8
2. RELATED WORK.....	11
2.1. CSA Systems	11
2.1.1. Related Work on CSA Systems.....	11
2.1.2. Surveys about CSA Systems	13
2.1.3. Comparison of Previous CSA Studies	14

2.1.4. Proposed CSA System Comparison with Previous CSA Studies.....	17
2.2. Focused Web Crawlers.....	18
2.2.1. Related Work about Web Crawlers	19
2.2.2. Related Work about Web Crawler Development	20
2.3. Sentiment Analysis.....	22
2.3.1. Related Work on Sentiment Analysis which also Utilize Non- Statistical Methods.....	22
2.3.2. Related Work on Sentiment Analysis which Utilize only Statistical Methods	27
2.4. Record Linkage	28
2.4.1. Related Work about General Record Linkage.....	29
2.4.2. Related Work about Product Matching Record Linkage.....	37
2.4.3. Surveys about Record Linkage.....	41
3. ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING	45
3.1. Standardization/Preprocessing Methods.....	45
3.1.1. Spelling Correction.....	45
3.1.2. Diacritics Transformation.....	45
3.1.3. Lower Case Conversion	46
3.1.4. Data Transformation.....	46
3.1.5. Punctuation and Non-ASCII Character Removal.....	46
3.2. Tokenization and Bag-of-Words Presentation	46
3.3. N-Grams	48
3.4. Weighting Schemes	48
3.4.1. Term Frequency Variants	49
3.4.2. Inverse Document Frequency Variants	49
3.4.3. Relevance Frequency.....	50
3.5. Distance Metrics.....	50
3.6. Classification	54

3.7. Clustering	56
3.7.1. Searching/Blocking	56
3.7.2. Hierarchical Agglomerative Clustering (HAC).....	56
3.8. Performance Evaluation Methods	61
3.8.1. Precision, Recall, Accuracy and F-measure	62
3.8.2. Adjusted Rand Index	63
3.9. Feature Selection	64
3.9.1. Entropy and Information Gain.....	65
3.9.2. Chi-Square	66
4. PROPOSED FOCUSED WEB CRAWLER SYSTEM	69
4.1. The Architecture of EcCrawler.....	69
4.2. Database Design of EcCrawler.....	70
4.3. Initialization.....	72
4.3.1. Thread Pool Initialization	72
4.3.2. Exception Handling	73
4.3.3. UI Independent Tasks.....	73
4.3.4. Timer-Based Polling Methodology	74
4.4. Main Thread	74
4.5. URL Handler	75
4.6. URL Crawler	79
4.7. Database Communication.....	86
4.8. Pages Processor	88
4.9. Collaboration and Resource Sharing Between Different Threads.....	91
4.10. Error Log System	91
4.11. Thread Count Management System	92
5. PROPOSED CLASSIFICATION SYSTEM for SENTIMENT ANALYSIS.....	95
5.1. Dataset of Sentiment Analysis.....	95
5.2. The Classification System	96
6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE ..	99

6.1. Matching Functions Used during Distance Computations	105
6.1.1. Price Feature Matching.....	106
6.1.2. Supervised Brand Filtering.....	106
6.1.3. Word Conflict Check.....	107
6.1.4. Product Code Detection.....	108
6.2. Dataset of Product Record Linkage.....	112
7. PROPOSED SEARCH ENGINE and USER INTERFACE	115
7.1. Search Engine	115
7.1.1. Most Related Products.....	115
7.1.2. Most Popular Products	117
7.2. Proposed User Interface	119
8. EXPERIMENTAL EVALUATION.....	125
8.1. Focused Web Crawler Development Experiments.....	125
8.1.1. Database Communication.....	128
8.1.2. Source Code Compression.....	131
8.1.3. Using Compression While Fetching Pages	134
8.1.4. Initialization of Thread Pool.....	137
8.1.5. Garbage Collector Mode Experiment.....	139
8.1.6. Concurrent Connections Limit Experiment.....	141
8.1.7. All Default Settings Experiment.....	143
8.2. Sentiment Analysis – Classification Experiments.....	145
8.2.1. Experimental Setup	145
8.2.2. Experimental Results and Discussions	147
8.2.3. Comparison of Previous Works with this Study.....	154
8.3. Record Linkage - Clustering Experiments	158
8.3.1. Experimental Setup	158
8.3.2. Experimental Results and Discussions	161
8.4. Product Search Engine Performance Experiments	172
8.4.1. Precision Experiments of Existing CSA Systems	173

8.4.2. Performance Experiments of Top Products in a Category Search....	174
9. CONCLUSIONS AND FUTURE WORK.....	177
9.1. Focused Web Crawler Development.....	177
9.2. Sentiment Analysis of E-Commerce Product Comments.....	177
9.3. Record Linkage Task of E-Commerce Products	178
9.4. Proposed CSA System.....	179
REFERENCES	181
CURRICULUM VITAE	203
APPENDIX	204



LIST OF TABLES	PAGE
Table 2.1. Comparison of previous CSA systems in the literature	15
Table 2.2. Comparison of our proposed CSA system with previously proposed CSA systems	17
Table 3.1. Vectoral representation example.....	47
Table 3.2. Variants of TF weighting.....	49
Table 3.3. Variants of IDF weighting	50
Table 3.4. Distance Metrics	51
Table 3.5. Clustering Linkage Criteria.....	58
Table 4.1. Statistical analysis of concurrent connection limit on E-commerce websites.....	85
Table 5.1. Statistics of all comments that are crawled by our focused crawler system	95
Table 6.1. Special string replacement list	101
Table 6.2. Semantic word replacement entire list.....	101
Table 6.3. An example of data preprocessing steps in record linkage task.....	102
Table 6.4. A list of conflicting words	108
Table 8.1. Experimental results for synthetic database connection	130
Table 8.2. Comparison of the compression methods	133
Table 8.3. The effect of compression enabled HTTP requests experiment	136
Table 8.4. Result of the thread pool initialization experiment.....	138
Table 8.5. Results of the garbage collector mode experiment	140
Table 8.6. Results for the concurrent connection limit experiment	142
Table 8.7. Results for all default settings experiment.....	144
Table 8.8. Best product comments classification results of each TF and IDF weighting metrics on our collected dataset.....	148
Table 8.9. N-gram preprocessing effect on Turkish E-commerce product comments sentiment analysis.....	149

Table 8.10. Number of features in the entire product comments dataset when N-gram method is applied.....	149
Table 8.11. List of the tested product comments classification configurations on the other datasets.....	155
Table 8.12. Sentiment analysis experimental results for English and Turkish movie reviews datasets	155
Table 8.13. Experimental results for English product reviews dataset (Blitzer, et al., 2007)	156
Table 8.14. Experimental results for Turkish product reviews dataset (Demirtas & Pechenizkiy, 2013).....	156
Table 8.15. Most promising configuration of the first stage experiments	163
Table 8.16. Experimental results of distance metric, linkage criteria and distance threshold for record linkage task.....	164
Table 8.17. Results of parameter experiments for the product code detection system	165
Table 8.18. Results of the final stage parameter experiments for record linkage task	166
Table 8.19. Experimental results of the macro-average dataset clustering experiments.....	168
Table 8.20. Changed configurations after re-training on the dataset of new clustering experiments	171
Table 8.21. Results of new clustering experiments.....	172
Table 8.22. Precision experiments of top CSA services in Turkey.....	173

LIST OF FIGURES	PAGE
Figure 3.1. Possible example solutions for a Linear SVM (Team, 2016)	55
Figure 3.2. Optimal solution for a Linear SVM where solid color filled shapes are support vectors (Team, 2016).....	55
Figure 3.3. An example for Agglomerative and Divisive Hierarchical Clustering (Kassambara, 2015).....	57
Figure 3.4. Graphical representation of Single-Linkage, Complete-Linkage, Average Linkage, and Centroid Linkage methods (Veress, 2013)	59
Figure 3.5. Complete-Linkage example (Kassambara, 2015).....	59
Figure 3.6. Single-Linkage example (Kassambara, 2015)	60
Figure 3.7. Average Linkage example (Kassambara, 2015)	60
Figure 3.8. Wards Linkage example (Kassambara, 2015)	61
Figure 4.1. Flowchart of EcCrawler	70
Figure 4.2. The tables used in EcCrawler.....	71
Figure 4.3. Default max persistent connections per server settings for Mozilla Firefox version 44.0.2	83
Figure 6.1. An example of a cluster dendrogram (Kassambara, 2015)	100
Figure 6.2. Simplified flowchart of the proposed clustering system.....	105
Figure 6.3. Pseudo code of Product Code Matching Algorithm	111
Figure 7.1. Product search engine user interface.....	119
Figure 7.2. Screenshot of Search a Product results	120
Figure 7.3. Screenshot of how details of a product are presented to the users	121
Figure 7.4. Screenshot of how positive comments of a product are displayed ...	122
Figure 7.5. Screenshot of Search Most Popular results.....	123
Figure 8.1. Table and query used in the synthetic tests.....	130
Figure 8.2. Classification accuracy when top N% (0.1% - 10%) features are selected.....	151

Figure 8.3. Classification accuracy when top N% (10% - 97%) features are selected.....	151
Figure 8.4. Percentage of training documents that become empty when top N% feature selection is applied.....	152
Figure 8.5. Percentage of test documents that become empty when top N% feature selection is applied.....	152
Figure 8.6. An example of the settings that are used to calculate clustering evaluation by ELKI.....	160
Figure 8.7. An example clustering evaluation results generated by ELKI.....	161
Figure 8.8. The top result of Google “en iyi ekran kartı” (best graphic card) search	175
Figure 8.9. Google best graphic cards query.....	176

LIST OF CODE SNIPPETS	PAGE
Code Snippet 1. How to initialize the Thread Pool.....	206
Code Snippet 2. How to setup global exception handler and close handlers	206
Code Snippet 3. How to start new threads independently from the main thread	206
Code Snippet 4. How to start timer-based polling functions.....	206
Code Snippet 5. How to refresh the user interface without causing UI freeze .	207
Code Snippet 6. How to set Garbage Collector mode.....	207
Code Snippet 7. An example for usage of data structures and locking to ensure thread safety and data consistency	207
Code Snippet 8. A shortened version of EcCrawler's fetching function	208
Code Snippet 9. How to increase the maximum number of concurrent connections to the same host and improve the fetching performance of the crawler by tuning the network configuration.....	209
Code Snippet 10. A shortened version of batch command SQL query execution function of EcCrawler.....	210
Code Snippet 11. Examples for proper locking to provide thread safety.....	211
Code Snippet 12. An example for multi-thread management system that ensures responsiveness of the UI.....	212



LIST OF ABBREVIATIONS

E-commerce	: Electronic Commerce
BKM	: Interbank Card Center
CSA	: Comparison Shopping Agent
EcCrawler	: Electronic Commerce Crawler
URL	: Uniform Resource Locator
ICANN	: Internet Corporation for Assigned Names and Numbers
NLP	: Natural Language Processing
HAC	: Hierarchical Agglomerative Clustering
HTML	: HyperText Markup Language
HTTP	: Hypertext Transfer Protocol
SVM	: Support Vector Machine
TF	: Term Frequency
IDF	: Inverse Document Frequency
RF	: Relevance Frequency
WS	: Weighting Scheme
IG	: Information Gain
CHI2	: Chi-square
CD	: Compact Disc
API	: Application Programming Interface
DAML	: DARPA Agent Markup Language
OIL	: Ontology Inference Layer
XML	: Extensible Markup Language
I/O	: Input/Output
NB	: Naive Bayes
POS	: Part-of-Speech
BOW	: Bag-of-Words
K-NN	: K-Nearest Neighbors

UPC	: Universal Product Code
SNM	: Sorted Neighborhood Method
RDF	: Resource Description Framework
AGNES	: Agglomerative Nesting
DIANA	: Divisive Analysis
TP	: True Positives
FP	: False Positives
TN	: True Negatives
FN	: False Negatives
UI	: User Interface
WPF	: Windows Presentation Foundation
GC	: Garbage Collector
DOM	: Document Object Model
JSON	: JavaScript Object Notation

1. INTRODUCTION

For the last 2 decades, the advances in Internet technology brought many commercial opportunities along with it, both in Turkey and in the world. Interactive product selling can be said as one of the biggest opportunity. The volume of interactive shopping is exponentially increasing and it has become a common part of the trade business. The interactive shopping of products over electronic systems such as Internet is described as E-commerce (Electronic Commerce) (Chen & Dhillon, 2003). The E-commerce shopping is becoming an irrevocable part of our lives as each year passes. With the emerging new technologies, advances in the technology and the increase in the technological education level of the consumers, it is now much easier to do E-shopping than before. According to the Turkey's official BKM (Interbank Card Center) statistics (BKM, 2016), the electronic shopping volume in Turkey has increased to 50,682,910,000 TL in 2015 from 16,553,790,000 TL in 2011. This massive increase in transaction volume attracts many businesses to start operating in E-commerce area. Thus, many new E-commerce websites are starting to operate continuously. Due to the stiff competition among E-commerce businesses, many also being have to halt their operation. Additionally, because of the competition, the E-commerce websites constantly update their prices and make discounts or campaigns. Therefore, it is quite not possible for average consumers to keep track of E-commerce websites when going to make an online purchase. To provide best prices for an online product and the other features of the product to the consumers, CSA (Comparison Shopping Agent) systems are developed (Pathak, 2010). These systems collect the necessary information from E-commerce websites and provide the processed structured information to the consumers. Because of these stated reasons, in this study we present a full scale CSA system for Turkish E-commerce websites which is named as EcCrawler (Electronic Commerce Crawler).

1.1. Preliminaries**1.1.1. CSA Systems**

An advanced CSA system would include, helping users to make decision on which product to buy, finding important information such as user feedbacks or specification on those products, price comparison to show cheapest products, exploring to find and list all of the new products and stores, frequent rechecking to not miss any offers and discounts (Doorenbos, et al., 1997). Providing all of these features requires complex phases that can be listed as below:

- i. A specialized focused web crawler has to be developed in order to fully crawl targeted E-commerce websites. The E-commerce websites generally tends to have badly constructed URL (Uniform Resource Locator) structure that makes it difficult to efficiently (e.g., infinite number of duplicate pages) crawl these E-commerce websites. Also there can be spider traps or other restrictions to prevent computer programs crawling the website. In order to solve these problems, the developed crawler need to be efficient, robust and error tolerant.
- ii. Proper wrapper generation is required to extract necessary product information from crawled E-commerce websites. It has to be robust and error tolerant.
- iii. Well planned database design in order to retrieve/save necessary data. Since it has to be scalable, an inefficiently designed database can be a great burden to scalability.
- iv. A high accuracy sentiment analysis system to evaluate customer feedbacks on products is required.
- v. Efficient and high performance record linkage system that will cluster same products from different E-commerce websites into the same cluster is a must. We propose that the record linkage is the most important part of the CSA systems because a good CSA system has to be able to cluster

same products from different E-commerce websites with at least 85% precision (Kannan, et al., 2011a).

- vi. An internal searching mechanism that processes user query and generate results has to be developed.
- vii. A user interface that provides searching interface and presents these results to users is necessary.

1.1.2. Wrappers

In today's web, there are so many different information sources. Thus, for the agents, it is not possible to integrate information from multiple sources with a singular information retrieval procedure. To solve this problem, a translator system is required to convert raw data into the structured data for each sources. These systems are referred as wrappers (Kushmerick, 1997). Wrappers can be manually, semi-automatically, and automatically generated.

1.1.3. Focused Web Crawlers

Web crawlers are software systems that are used to collect information from websites. Their main task is fetching websites, processing fetched source code and extracting new target hyperlinks to crawl. General web crawlers start with root URLs and continue until crawling all of the hyperlinks that they can find. However, this task is tedious and not doable for average or even small scale applications. Only a few large-scale commercial general search engines (e.g., Google, Bing, Yahoo, Yandex, and so on) can cope with the challenges and the massiveness of the entire web and keep their index fresh. They need to use many different sources (e.g., being a member of ICANN (Internet Corporation for Assigned Names and Numbers) and getting the list of newly registered domains, and so on) to discover the new links and keep recrawling their existing URLs just to maintain freshness. Because of these very reasons, developing focused web crawlers are much more feasible and commonly practiced.

Focused web crawlers are specialized versions of general web crawlers that crawl only certain topics or certain websites (Chakrabarti, et al., 1999). Even though they are much smaller scale than general web crawlers, still many challenges and tough tasks await the developers who are going to build focused web crawlers (Gupta, 2012). In this thesis, it is aimed to share experiences that are gained from the development of a full-scale, object oriented, robust, and fast focused web crawler that uses the latest technologies with multithreaded implementation and has a very high success rate by employing hand-crafted rules. The development is made in Visual C# (Microsoft, 2016) programming language based on the WPF (Microsoft, 2016) programming model by using .NET 4.5 (Microsoft, 2016) framework and x64 platform. MS-SQL Server 2014 (Microsoft, 2016) is used for database management. For programming, Visual Studio 2013 IDE (Microsoft, 2016) is used.

1.1.4. Sentiment Analysis

With the emerging new technologies, today, electronic platforms such as internet blogs, E-commerce websites, digital newspapers, Facebook, Twitter, online forums, etc. are crucial for our lives. People not only read these platforms anymore but also leave their opinions about the related subjects. The analysis of these massive opinions has become a major interest recently in the NLP (Natural Language Processing) field. Sentiment analysis or opinion mining is the computational study of people's opinions, appraisals, attitudes, and emotions toward entities, individuals, issues, events, topics and their attributes (Liu & Zhang, 2012). In this study, sentiment analysis is used to determine contextual polarity of the user comments which are made for E-commerce products. Thus, we propose that user opinions about E-commerce products do matter and need to be included in any good CSA system. Moreover, sentiment analysis of morphologically rich languages such as Turkish still has not got much interest in the literature (Kaya, et

al., 2012). Because of these two reasons we have conducted research on this subject.

1.1.5. Record Linkage

Record linkage is the process of trying to determine which records from two or more different data sources represent the same real-world entity (Elfeky, et al., 2002). Therefore, in this study, we define our clustering task to determine the same products from different E-commerce sites problem. Our proposed novel system uses modified HAC (Hierarchical Agglomerative Clustering) algorithm to be incremental/dynamic and has high success rate on the record linkage task of Turkish E-commerce products.

1.2. Summary of the Contributions

1.2.1. Contributions in the Focused Web Crawler Area

The challenges when developing a focused web crawler for which possible solutions are proposed in this study are listed as below:

- i. Providing communication between user interface (i.e., the main thread) and background worker threads,
- ii. Maintenance of responsive user interfaces while running thousands of threads,
- iii. Collaboration and resource sharing between different threads,
- iv. Unexpected error handling,
- v. Maintenance of URL repository,
- vi. Robust web page fetching,
- vii. Keeping logs for errors,
- viii. Providing optimized database access for performance improvement,
- ix. Being able to process erroneous HTML (HyperText Markup Language) source,

- x. Handling different HTML source page structures for data extraction from different websites,
- xi. Compressing HTML source files to save disk space,
- xii. Saving bandwidth by HTTP (Hypertext Transfer Protocol) compression,
- xiii. Management of the number of threads that run simultaneously.

Since focused web crawlers are a specialized version of general web crawlers, many of the above challenges are also the concern of general web crawlers as well. Therefore, we believe that this study can also help general web crawler developers too.

1.2.2. Contributions in the Sentiment Analysis Area

In this study, we present results of extensive experimental tests conducted by using a SVM (Support Vector Machine) classifier to do sentiment analysis on Turkish product reviews. We use LIBSVM (Chang & Lin, 2011) framework for the SVM-based classification task. Furthermore, we present a detailed comparison table which includes results of previously related works in the literature and results of our experiments on the datasets of these related works. The contributions of this study are summarized as below:

- i. Detailed empirical evaluation of SVM types, kernel types, kernel parameters, and shrinking feature of SVMs are made.
- ii. Numerous document vector computation techniques that are TF (Term Frequency), IDF (Inverse Document Frequency), TF*IDF, RF (Relevance Frequency), TF*RF weighting schemes with state-of-the-art TF and IDF variances are experimented over several datasets.
- iii. Unigram word + different character N-gram word tokenization approaches and weight vectors normalizations are evaluated.

- iv. Information Gain (IG) and Chi-square (CHI2) feature selection methods are thoroughly evaluated.
- v. Preprocessed versions of all of the experimented datasets are made publicly available to provide an objective comparison with the results of this study and future works that will be done by other researchers. Moreover, a new meticulously prepared Turkish product reviews dataset is now available to researchers.
- vi. In earlier studies, usually expensive to compose dictionaries or computationally complex linguistic methodologies are employed to obtain high success rates. However, in this study, we obtain high classification accuracy by solely using the correct combination of the NLP methods and text analysis techniques and we list these techniques in this study to help researchers for their future studies.
- vii. To our knowledge this is the first study which evaluates numerous document vector computation methods for sentiment analysis of Turkish product reviews.

1.2.3. Contributions in the Record Linkage Area

The contributions of this study for the record linkage problem for Turkish E-commerce products are summarized as below:

- i. We propose a new incremental/dynamic HAC system for the record-linkage problem of E-commerce products.
- ii. A novel product matching/filtering system which improves the success of the clustering task significantly is proposed.
- iii. A new product code/identifier detection system for the E-commerce products is proposed.
- iv. We propose an evaluation approach by using an open source clustering framework for the record-linkage area. The success rate computation for

clustering task highly changes due to the used algorithm and its implementation. Our proposed approach provides objectively comparable results for the future researchers.

- v. Very detailed experimental results for the state-of-the-art algorithms are presented.
- vi. To our best knowledge, we publish the first publicly available E-commerce record-linkage dataset. Additionally, we provide very detailed preprocessed versions of our dataset. Thus, we make it possible that the future studies can precisely evaluate their results and compare them with our results.

1.2.4. Contributions in the Product Search Engine Area

The contributions of our proposed product search engine can be summarized as below:

- i. Higher precision for product searching by minimizing necessary human laboring and utilizing advanced record linkage system.
- ii. Providing accumulated product comments from different sources to the users in an appealing way. Thus, they can have more idea about the products before deciding which one to purchase.
- iii. Better categorical product search system than the existing similar systems by utilizing polarity of the product comments and the number of stores they are sold.

1.3. Organization of the Thesis

We thoroughly present the related work in the Section 2 where we provide information on CSAs, focused web crawlers, sentiment analysis, and record linkage task. In Section 3 we provide information about utilized algorithms and methodologies where we cover standardization methods, document vectorization,

N-grams, weighting schemes, distance metrics, classification task, clustering task, performance measurement algorithm for classification and clustering tasks, and feature selection methods for sentiment analysis. In Section 4, we present our proposed focused web crawler system. In Section 5 and Section 6, we propose our classification and clustering systems for the sentiment analysis and record linkage tasks respectively. In Section 7, proposed search engine system and the user interface of the entire CSA system are presented. In Section 8, very detailed experimental results of the focused web crawler development process, record linkage task of the E-commerce products, sentiment analysis of the E-commerce product reviews, and the performance analysis of the search engine are presented. In the last section, we conclude the study and discuss some possible future works.



2. RELATED WORK

2.1. CSA Systems

Since CSA systems have become a great commercial asset in today's world, there is not much published literature work recently about CSA systems. Majority of the related work utilizes internal search systems of the E-commerce websites and they do not have any kind of record linkage system like the proposed CSA system in this thesis study. To our best knowledge, there is not a single closely related work to our study.

2.1.1. Related Work on CSA Systems

Widely recognized first comparison agent is BargainFinder (Krulwich, 1996). BargainFinder is a focused CSA system that scans only online CD (Compact Disc) stores. It uses internal search system of the E-commerce websites in order to display results to the users. When a user submits a query, it transforms query into the internal searching structure of the online CD stores and then submits query to the registered online stores in its system. When the results are returned from the online CD stores, it integrates results and then displays to the users. It does not utilize any record linkage system.

Doorenbos and colleagues (1997) have proposed a CSA system named as ShopBot that generates automatic wrappers to extract information from online shopping websites. ShopBot works as two phases. In the first phase, ShopBot learns the internal searching system of the E-commerce websites and generates wrappers automatically to conduct searches. In the second phase, ShopBot queries the E-commerce stores and retrieves the results. Wrapper generation happens offline and need to be done only one time for each shopping website. After offline learner generates wrappers for the registered shopping websites, it takes queries from the users to send shopping sites and then display the returned results to the users. Main disadvantage of ShopBot is to have strong biases. ShopBot assumes

that shopping websites have searchable index (navigation regularity), display all products in the same format (uniformity regularity), and have the same usage of whitespaces (vertical separation regularity). All these biases make it impracticable for today's technology and ShopBot does not employ any kind of record linkage system to detect the same product entities from different sites.

Yang and colleagues (2000a), Yang and colleagues (2000b), Santos and colleagues (2001), and Pivk and Gams (2002) have also proposed CSA systems that have the same working principle as ShopBot (Doorenbos, et al., 1997).

Rather than how to design a wrapper generation system for a shopping bot like the previous studies on shopping bot design, Montgomery and colleagues (2004) have proposed approaches to improve shopping bot utilization for better consumer experience. Their considered CSA system works by using the same principle as ShopBot (Doorenbos, et al., 1997) has. When a consumer wants to search for an item among E-commerce sites, the considered shopping bot makes live queries over these E-commerce sites and return results. The major payoff of this approach is the delay. Querying each one of the E-commerce sites is taking huge time and this is leading consumers to be impatient and leave before the agent process E-commerce websites. So they have worked on, how many E-commerce sites to query, how long to wait for responses, and how many of these responses is presented to provide better experience to the consumers. Our proposed system does not have such problem since no online querying is employed on the E-commerce websites.

Lee and colleagues (2003) have proposed an E-commerce system which uses semantic web techniques that are readable by machines. Therefore, these E-commerce websites can support CSA agents. Their proposed system uses DAML (DARPA Agent Markup Language) meta-data, DAML+OIL (Ontology Inference Layer) ontology and reasoning engine with JESS (A rule engine for the Java platform) & DAMLJessKB (Description logic reasoner for the DARPA) in order to

mark up shopping web pages with semantic web languages that would make them machine readable.

Huang and Tsai (2011) have proposed a cross-language CSA system that utilizes the XML (Extensible Markup Language) data and the HTML source of E-Commerce pages. Main focus of their study is designing a system that will semantically evaluate both the query and the products. Thus, users can multilingually search for the products they are looking for. Their system utilizes APIs (Application Programming Interfaces) of Google Dictionary and Wikipedia to perform the automatic translation.

2.1.2. Surveys about CSA Systems

Guttman and colleagues (1999) have surveyed several agent-mediated E-commerce systems by describing their roles in the context of a Consumer Buying Behavior Model. They have compared existing software agents from the following concepts: (1) Requirement Analysis (this stage characterizes the consumer and thus becomes aware of some unsatisfied need), (2) Product Brokering (this stage comprises the retrieval of information to help to determine what to buy), (3) Merchant Brokering (who to buy from), (4) Negotiation (how to determine the terms of the transaction), (5) Purchase and Delivery, and (6) Product Service and Evaluation. In this thesis study, “Product Brokering”, “Merchant Brokering”, and “Product Service and Evaluation” areas are worked on.

Sadeddin and colleagues (2007) have made empirical performance analysis of nine existing CSA systems. They have used 40 books, 20 CDs, and 20 DVDs to query CSA systems and analyze the returned results to determine effectiveness based on the accuracy, consistency, and repeatability of the recommendations by using product price attribute as a key measure. They have concluded that there is no best shopping bot available. According to their experiments, most of the CSA systems offer very limited product information to the end users and all of the CSA systems often present inaccurate information in terms of the actual product price or

the product availability. These findings also prove the importance of this thesis study.

Wan (2009) have made a comprehensive analysis of existing CSA systems. They have covered “How to design a good shopbot”, “How shoppers use the comparison-shopping services”, “What is the strategic use of comparison-shopping as a new channel by online vendors”, “What is the impact of comparison-shopping on existing price equilibrium and electronic market structures” topics empirically.

Pathak (2010) have made a survey of CSA based decision support systems. Focus of their study is the contemporary literature about the CSAs to analyze them in the context of decision support systems. They have proposed that a typical decision support system should have the following four components: data, models, interfaces, and user specific customization to provide comprehensive decision support.

2.1.3. Comparison of Previous CSA Studies

Table 2.1 includes a comparison for the previous CSA studies in the literature with our study.

Table 2.1. Comparison of previous CSA systems in the literature

Name of the Study	Coverage	Focus of the Study	Methodology	Algorithms Used
Bargainfinder (Krulwich, 1996)	Online CD stores	Wrapper generation	By exploiting internal search system of E-commerce web sites	No algorithm is described
Shopbot (Doorenbos, et al., 1997)	Online software and CD vendors	Wrapper generation	By exploiting internal search system of E-commerce web sites	Hidden learner algorithm that exploits success and failure pages of E-commerce sites
Morpheus (Yang, et al., 2000b)	Online shopping web sites	Wrapper generation	By exploiting internal search system of E-commerce web sites	Searching for a sequence of repeated patterns (exploiting uniform data presentation)
No new system but evaluation of Wherebuy (dos Santos, et al., 2001)	Online CDs, DVDs, books	Case study, architecture proposal	By exploiting internal search system of E-commerce web sites	User interface, product catalogs (e.g., brand), meta-search engine, general library (e.g., HTTP handlers)
Shina (Pivk & Gams, 2002)	Online shopping sites	Wrapper generation	By exploiting internal search system of E-commerce web sites	Semi-supervised wrapper generation by help of the user input

Table 2.1. Comparison of previous CSA systems in the literature (Continued)

No CSA design (Montgomery, et al., 2004)	Online shopping sites	Focus on the operational decisions made by a CSA system	By exploiting internal search system of E-commerce web sites	Improving utilization of CSAs which are based on internal search systems of web sites (e.g., how many stores to query)
A system that will support CSA systems (Lee, et al., 2003)	Online shopping sites	Ontology for E-commerce sites	By using markup languages to make E-commerce sites machine readable	DAML, DAML+OIL, JESS, DAMLJessKB
Webshopper+ (Huang & Tsai, 2011)	Online shopping sites	Constructing a multilingual ontology	By utilizing the shopping sites provided XML data and the source of the HTML pages	Google dictionary API and the Wikipedia API (no information about how to collect, parse, cluster, organize data)
Our Proposed CSA System	Online shopping sites	Record linkage on E-commerce products + Sentiment analysis of user comments on E-commerce products	By utilizing proposed clustering system on product titles and classification system on product comments	Incremental HAC + SVM

2.1.4. Proposed CSA System Comparison with Previous CSA Studies

The similarities and differences between previously proposed CSA systems and our proposed CSA system are presented in Table 2.2.

Table 2.2. Comparison of our proposed CSA system with previously proposed CSA systems

Our Proposed CSA System	Previously Proposed CSA Systems
Our system does record linkage on E-commerce products	Previous systems utilizes internal searching systems of E-commerce websites. Thus, they do not have any record linkage
Our system constructs wrappers semi-automatically to obtain very high success rates on extraction of the necessary structured data	Previous systems focuses mainly on full or more semi-automatic wrapper generation system than our system with the cost of success loss
Our system does not make any online queries to E-commerce websites when a user search for a product	Previous systems do online querying from the E-commerce websites. Thus, they work extremely slowly and inefficiently when compared to our system
Our system provides categorical popular product search system with the offline data by utilizing semantic evaluation of user comments and statistics of the products	Previous systems cannot provide such information since they do not contain offline E-commerce data and they do not employ any semantic evaluation of user comments
Our system semantically evaluates user comments to show to users and use in product popularity system	Previous systems do not have such semantic evaluation feature
Our system does not make any categorical selection on E-commerce products. It works on every category of products	Most of the previous systems are designed to work on only certain category of products (e.g., only Online CD stores)

2.2. Focused Web Crawlers

In the literature, the majority of the related works are about how to design a general web crawler (Heydon & Najork, 1999; Shkapenyuk & Suel, 2002; Boldi, et al., 2004; Gomes & Silva, 2008) or how to decide more relevant pages for a focused web crawler to reduce resource requirements such as space, bandwidth, computation power, etc., in the crawling task (Shkapenyuk & Suel, 2002; Yohanes, et al., 2011; Liu & Milios, 2012; Bedi, et al., 2012; Maimunah, et al., 2012; Liu & Du, 2012; Papavassiliou, et al., 2013; Li, et al., 2013; Kumar & Vig, 2013; Uzun, et al., 2014; Dahiwal, et al., 2014; Nisha & Sundareswari, 2015). To our knowledge, there is no study that explains the challenges and the possible solutions for developing a (focused) web crawler from developers' perspective by providing implementation detail and being up-to-date. Additionally, the web has become so enormous and spammy that most of the previously proposed approaches may fail today. Google (2016) displays statistical data about spam pages on the web and the domains that are penalized by Google manually. These statistics clearly show an exponential increase of spam. Furthermore, nowadays spammers are able to generate more sense making spun content¹ by using machine learning algorithms. They are also able to build highly related link structures. Thus, depending solely upon link relations, semantic relevance or content analysis can lead to a failure. Other than the multi-million budget having companies, it is not very possible to scan the entire web to determine the authoritativeness of websites or obtain enormous golden data to weigh them accurately. For the reasons outlined previously, if there is a need for a web crawler that requires high success rate, we propose that developing a task oriented, new and hand-crafted focused crawler may be the best solution and we do so in this thesis study.

Our study is different from the previous studies that have been made in this area such that we provide source codes to solve challenging problems that may be

¹ Article spinning – Wikipedia, https://en.wikipedia.org/wiki/Article_spinning

encountered during crawler implementation. Additionally, we provide extensive empirical performance analysis for the proposed solutions.

2.2.1. Related Work about Web Crawlers

Edwards and colleagues (2001) have developed a fully distributed general web crawler model which mainly focuses on an efficient incremental crawling strategy.

Gomes and Silva (2008) have proposed a generalized web crawler and discussed some of the challenges they have encountered. Java programming language is used for implementing the crawler but no programming language related problems, their solutions, and optimizations are mentioned. Their main focus is more about explaining problems that are encountered while crawling the web and the structure of their crawler. Olston and Najork (2010) have made an extensive review of the web crawling literature. They have covered the majority of the web crawling topics such as crawler architecture, crawl ordering, avoiding undesirable content, and so on.

Yohanes and colleagues (2011) have proposed a genetic algorithm by using methodology that employs adaptive and heuristic methods to improve the relevancy of the crawled pages. Liu and Millos (2012) have proposed a framework that calculates the distance of the newly discovered URLs from the topic by using Maximum Entropy Markov Model and Linearchain Conditional Random Field probabilistic models by exploiting multiple features such as anchor text (the visible and clickable part of the hyperlink), etc. Bedi and colleagues (2012) have proposed the Dynamic Semantic Relevance system, which uses terms and links to improve crawling efficiency and precision of focused web crawlers. Maimunah and colleagues (2012) have proposed a system that uses relations between parent, sibling, target, child and spouse (which links to child) documents to improve precision and recall of focused web crawling. Liu and Du (2012) have developed an approach that uses Semantic Similarity Vector Space Model by utilizing

TF*IDF and semantic similarities to calculate relevancy of the uncrawled URLs. Papavassiliou and colleagues (2013) have developed a system for harvesting topic specific data from the web for both monolingual and bilingual pages. Li and colleagues (2013) have proposed a framework that uses divide and conquer strategy by using double step page classifier, link evaluation, when to stop crawling strategy, etc. to enhance the focused crawling success. Kumar and Vig (2013) have developed an algorithm that is based on TF*IDF which improves the precision of topic targeted crawling in consecutive crawling phases. Uzun and colleagues (2014) have used Decision Tree Learning² to automatically determine useful parts of the web pages. Different blocks of the web pages are identified by using HTML tags in order to improve the efficiency of web crawling and data extraction. Dahiwalé and colleagues (2014) have proposed a crawler that utilizes link and content-based evaluation to improve the precision of focused crawling. Nisha and Sundareswari (2015) have proposed an approach to improve the selection of seed URLs by using user interest ontology.

Such related works list some efficient crawling strategies and show that systems can be extended, however, our aim in this study is different. In this research, our aim is to start crawling from a static root web site, and crawl all the pages except particular ones (e.g., blocked by robots protocol (Koster, 2007) or ignored by manually crafted rules, etc.) from the root domain with high success rate. To achieve this goal, we propose practical implementation based solution methods which involve fine tuning of some .NET framework based parameters, and we show the performance of our proposed solutions experimentally.

2.2.2. Related Work about Web Crawler Development

Heydon and Najork (1999) have proposed a highly-scalable general web crawler system and discussed the fundamental problems of a general web crawler

² Decision tree learning – Wikipedia,
https://en.wikipedia.org/wiki/Decision_tree_learning

development. Their proposed system has been developed in Java programming language. However, implementation details to solve problems encountered during the crawler design are not provided. Shkapenyuk and Suel (2002) have presented a system architecture of a distributed web crawler that runs on multiple connected servers. They have discussed I/O (Input/Output) performance, network efficiency, solutions for performance bottlenecks and how to obtain high performance when bulk crawling. They have used C++ and Python programming languages to develop the crawler. However, implementation details and solutions for coding related problems and tunings are not discussed.

Boldi and colleagues (2004) have proposed a scalable web crawler called as UbiCrawler which is similar to our study. In their study, every task is tried to be decentralized as much as possible. Their system is implemented by using Java programming language. However, details about how to improve programming language related performance issues are not discussed. Also, UbiCrawler does not guarantee 100% prevention from duplicate crawling, and the system may have problems to achieve continuous crawling when it is restarted or at total application crash. Moreover, how to distribute crawling hosts to the agents are not explained in detail. Due to their system design, even distribution of the workload among agents is difficult to achieve. The number of the URLs among the agents is assumed to be evenly distributed during the big crawls. But, our experiments indicate that this assumption may not be valid every time. Furthermore, due to the decentralization of the agents, they cannot communicate with the each other to access the disk, and this may cause a restriction to be put on the number of active agents to prevent hard drive from speed throttling. In EcCrawler, on the other hand, every agent is centrally controlled. Therefore, distribution of the workload among agents is highly achieved and the crashes or the application restarts do not cause any problem for continuous crawling. EcCrawler's design, capability, features, and programming language that is used are highly different from UbiCrawler.

2.3. Sentiment Analysis

There has been much research in the opinion mining and sentiment analysis area recently. There are three recent major surveys made by Pang and Lee (2008), Liu and Zhang (2012) and Vinodhini and Chandrasekaran (2012).

In this study, we have obtained our best accuracies without depending on any dictionary-based methodology unlike the many of the previous works. Therefore, our methods can be applied to any dataset in any language without requiring complex algorithms and computations.

2.3.1. Related Work on Sentiment Analysis which also Utilize Non-Statistical Methods

Pang and colleagues (2002) have conducted classification experiments on a movie reviews dataset by using standard NB (Naive Bayes), Maximum Entropy, and SVM classification algorithms. N-gram based tokenization, POS (Part-of-Speech) (Brown, 1957) tagging, and frequency versus binary term weighting by using BOW (Bag-of-Words) methodology are compared. In BOW methodology, the text is split into words by space character and each word represent a feature (Sriram, 2010). According to their experiments, unigram presence method performs best.

Dave and colleagues (2003) have proposed a system to classify product reviews as negative or positive. They have employed substitutions which require extra human intervention as data-preprocessing. For instance, the names of the products are replaced with *productname* so that different product names can be treated equally. A computationally expensive linguistic parser to process words sentence by sentence is also used. Linguistic features such as WordNet (Miller, 1995), colocation, stemming and negation, are tested however; it is observed that all these features decrease the success rate in all tests except stemming which increases the success rate only in a single test.

In the later study, Pang and Lee (2004) have proposed a system to extract objectivity containing part of the reviews to decrease the size and complexity of the dataset so that the objective part is discarded as it is not useful when classifying sentences into their polarities. They have experimented with using standard SVM and NB algorithms to classify reviews into their polarities. The classification accuracy of their dataset is improved from 82.8% to 86.4% by using empirically determined parameters. Hu and Liu (2004) have proposed a system for mining opinionated product features from customer reviews. Additionally, they have classified opinionated sentences as positive or negative by using NLP techniques and have achieved average 84.20% accuracy in sentence polarity classification.

Wilson and colleagues (2005) have proposed a system to accomplish polarity classification in phrase-level by using subjectivity clues from a lexicon. They have used BoosTexter AdaBoost.HM classification algorithm with 28 features methodology and they have obtained best 65.7% accuracy in polarity classification while their baseline accuracy has been 61.7%. Their system first determines whether a sentence is neutral or polar then the latter sentences are classified as negative, positive, both or neutral.

Blitzer and colleagues (2007) have proposed a system for better Domain Adaptation to utilize in sentiment classification task. They have improved Structural Correspondence Learning algorithm; constructed and publicly released a product comments dataset which consists of four different product categories. They achieve an average 46% relative error reduction.

Wilson and colleagues (2009) have expanded their previously proposed system to accomplish phrase-level polarity classification by using subjectivity clues from an expanded lexicon. They have obtained their best success rates by using BoosTexter AdaBoost.MH classification algorithm with 32 features methodology. Their best-obtained accuracy is 76.5% when classifying sentences as polar or neutral. When polar classifying subjective sentences, they have obtained

best 83.2% accuracy on manually labeled subjective sentences and 65.9% accuracy on automatically labeled subjective sentences.

Prabowo and Thelwall (2009) have proposed a hybrid classification system in which firstly a Rule-Based Classifier, secondly a Statistics Based Classifier, thirdly a General Inquirer Based Classifier, and finally an SVM classifier are applied. If one classifier fails to classify the test data, the unclassified test data is transferred to the next classifier. However, their Statistics Based Classifier system utilizes public search engines' querying mechanism to obtain Closeness Measure. Thus, the system is not applicable to big datasets.

Eroğul (2009) have experimented on Turkish movie reviews by using techniques which utilize linguistic features of the Turkish language, such as POS tagging, word polarity or spell correction. Additionally, they have tested N-gram methodology and used Zemberek, which is an NLP tool for the Turkish language, in their experiments. They have failed to obtain any better results with their experiments when compared to their BOW methodology based baseline.

Jang and Shin (2010) have proposed chunked sentiment analysis system, which uses contextual shifters (negation shifters and flow shifters). Negation shifters change the semantic orientation of the words and flow shifters control the flow of the sentiment like "however" or "but". They have used a polarity dictionary and a subjectivity lexicon as well. Their proposed contextual shifters and chunking methodologies or polarity based weighting system have failed to improve the results statistically significant on the short movie reviews dataset. However, their proposed system achieves better results than baseline TF*IDF weighting scheme in the news corpora.

Arora and colleagues (2010) have proposed a system that utilizes Subgraph Mining and Genetic Programming based feature construction to improve the success of SVM-based sentiment classification task. They have obtained best 76.93% accuracy on the English movie reviews dataset (Pang & Lee, 2005). As a base score with just using unigrams, they have obtained 75.66% accuracy.

Yessenalina and colleagues (2010) have proposed a system that automatically generates rationales (a portion of the document that is more relevant to its class) to reinforce polarity classification of the documents. Then, these automatically generated rationales are used to generate additional training data to improve the accuracy of the classification task.

Li and Liu (2012) have proposed a clustering system which uses K-means clustering algorithm with a combination of TF*IDF weighting scheme, several preprocessing techniques, feature selection and term scoring by the help of the WordNet for polarity classification problem of documents.

Kaya and colleagues (2012) have carried out sentiment classification experiments on political news from Turkish news sites. NB, Maximum Entropy, SVM and character based N-gram Language Model classifiers by using Lingpipe DynamicLMClassifier are used in experiments. The best accuracies that are observed are around 75% with empirically determined parameters.

Boynukalın (2012) have proposed a framework for emotion analysis in the Turkish language. They have combined machine learning algorithms with NLP methodologies and tools for sentiment analysis. NB and SVM classification algorithms are tested with punctuations removal, stop words removal, proper names removal, spelling correction, stemming, negation, POS tagging, and word level N-grams. They have classified sentences of Turkish fairy tales dataset into five opinions (none, sad, anger, fear, and joy) and have obtained 74.5% accuracy by using Complement Naive Bayes classifier and 69.42% accuracy by using an SVM classifier.

Seker and Al-Naami (2013) have experimented on a massive Turkish comments dataset. They have employed user assigned scores for evaluation. SVM, K-NN (K-Nearest Neighbors) and C4.5 algorithms are used for the classification task. As a weighting scheme TF*IDF, and for feature selection IG are used. Each algorithm classifies each test sample separately, and the most voted class is chosen as class label. Their system obtains approximately 55% F-measure.

Aytekin (2013) have proposed a system to classify comments that belong to products and services. The proposed system assigns positive and negative polarities to the words in the Turkish language automatically by utilizing translations from an English sentiment dictionary. The system uses NB algorithm for classification, utilizes the generated Turkish sentiment dictionary and obtains 72.71% accuracy.

Demirtas and Pechenizkiy (2013) have proposed a classification system that utilizes machine translation to improve classification accuracy by co-training. They have used NB algorithm for this purpose and obtained approximately 79% accuracy as the best on the English movie reviews dataset (Pang & Lee, 2005) and nearly 83% accuracy as the best on the Turkish movie reviews dataset by the proposed co-training classification system.

Akba and colleagues (2014) have evaluated different feature selection methods on a Turkish movie reviews dataset. They have employed different NLP tools and preprocessing techniques as well. They have obtained 83.9% F-measure by using an SVM classifier when binary classifying (positive or negative) the dataset. Both CHI2 and IG based feature selection methods achieve the same F-measure in polarity classification. They also have tested three categories (positive, negative and neutral) classification by using an SVM and an NB classifier and obtained best 63.3% F-measure. SVM classifier performs better than NB classifier.

Yıldırım and colleagues (2015) have experimented NLP layers on Turkish sentiment analysis. Their dataset has been collected from Twitter in the telecommunication domain, and they have processed the dataset both automatically and manually to have high confidence data. The effects of normalization, preprocessing, stemming, morphological analyses and POS tagging are tested and 73%-79% accuracy on the dataset with about 6% relative improvement by using empirically determined parameters are obtained.

Dehkharghani and colleagues (2015) have semi-automatically generated a Turkish polarity resource, SentiTurkNet, and experimented on Turkish movie

reviews. They have used three classifiers that are Logistic Regression, Feedforward Neural Networks, and SVM with sequential minimal optimization algorithm and obtained 91.11% accuracy as the best when classifying synsets into the three polarities. In this test, they have compared their classification results with manually classified polarities. In the final test, a Turkish movie reviews dataset is classified as positive, negative or objective and the highest accuracy achieved is 66.77% by using the Logistic Regression classifier.

2.3.2. Related Work on Sentiment Analysis which Utilize only Statistical Methods

O’Keefe and Koprinska (2009) have empirically evaluated the performance of different weighting metrics and feature selection methods combination by using SVM and NB classifiers. They also have proposed several new feature weighting and feature selection metrics as well. However, their study does not include the Turkish language.

Martineau and Finin (2009) have proposed a modified version of the TF*IDF that is named as Delta TF*IDF weighting system to improve the accuracy of the classification.

Paltoglou and Thelwall (2010) have made a thorough empirical analysis on weighting scores used in polarity classification task. They have experimented on publicly available datasets and obtained very high accuracies when compared to original state-of-the-art works done on those datasets. On the English movie reviews dataset (Pang & Lee, 2004) Paltoglou and Thelwall (2010) have obtained 96.90% accuracy however in (Pang & Lee, 2004) 87.2% accuracy is observed; for product reviews dataset (Blitzer, et al., 2007) 96.40% accuracy is computed by Paltoglou and Thelwall (2010), on the other hands, from 82.75% to 87.90% accuracies are obtained by Blitzer and colleagues (2007). However, in our study, as it is observed in (Blitzer, et al., 2007), we have failed to obtain such high success rates with the proposed metrics in (Paltoglou & Thelwall, 2010).

2.4. Record Linkage

Grouping same entities from different sources into the same cluster is known as record linkage (Fellegi & Sunter, 1969; Elfeky, et al., 2002; Jin, et al., 2003; Bilenko, et al., 2006), deduplication (Sarawagi & Bhamidipaty, 2002; Christen, 2012), name matching (Cohen, et al., 2003), data integration (Cohen W. W., 1998), data cleaning (Florescuand, 2000; Rahm & Do, 2000; Chaudhuri, et al., 2003), duplicate detection (Monge & Elkan, 1997; Bilenko & Mooney, 2003b; Bilenko & Mooney, 2003a; Elmagarmid, et al., 2007), information integration (Aizawa & Oyama, 2005), entity resolution (Bhattacharya & Getoor, 2006; Bhattacharya & Getoor, 2007; Christen, et al., 2009; Whang, et al., 2009), reference reconciliation (Dong, et al., 2005), merge/purge problem (Hernández & Stolfo, 1995), object identification (Tejada, et al., 2002; Singla & Domingos, 2005), entity matching (Shen, et al., 2005), identity uncertainty (Pasula, et al., 2002), reference matching (McCallum, et al., 2000), hardening soft information sources (Cohen, et al., 2000), and robust reading (Li, et al., 2004) in the literature and has been widely researched. However, as can also be seen from the different naming, there are not area standard definition nor algorithms for this task. It requires domain-specific algorithms and methodologies in most cases for obtaining high success rates (Tejada, et al., 2002; Jin, et al., 2003; Yan, et al., 2007; Köpcke & Rahm, 2010a; Christen, 2012). Thus, we propose that each domain in record linkage is unique. Furthermore, many of the previous works are done on a single category entities. For example, only on restaurants or only on authors. However, our study targets infinite number of categories such as video cameras, led TVs, smartphones, hard disks, printers, fans, grass mowers, perfumes and so on.

According to the Hastie and colleagues (2009), the dissimilarity measure choice affects the performance of clustering task more than the choice of the clustering algorithm and the choice of dissimilarity measure highly depends on the specific knowledge of the targeted domain. Even though this proposal is not correct for every domain in clustering area, our experiments also verify that specific

domain knowledge is a must to choose a correct algorithm and the dissimilarity measure. Thus, it is not possible for a single clustering system to be successful at every clustering domain. Because of this reason, we propose that a specific clustering system for each targeted domain need to be developed. By this proposal, our study is unique in the literature because to our knowledge, there is not any identical targeted domain area study.

2.4.1. Related Work about General Record Linkage

Elfeky and colleagues (2002) have presented a new tool for record linkage problem along with three different machine learning methodology for the record linkage problem. Their first proposed method is based on a classifier which requires trained data, the second method is based on an unsupervised clustering, and the final method is a combination of both classifier and unsupervised clustering based hybrid model. In their hybrid model, first, a small training dataset is composed by employing an unsupervised clustering algorithm to a small subset of the dataset. Then composed training dataset is utilized to train the classifier based model. They have conducted experiments on both synthetic and real-world datasets. Their real world dataset is similar to the dataset in our study. Their hybrid model is applied on Amazon products dataset for deduplication. However, since a single data source is used, they have heavily exploited the structure of the data source such as using blocking on categories and even on sub-categories which is not doable in our research. Even with this heavy exploiting, they have obtained 79.8% accuracy by employing unsupervised clustering method and 76.5% accuracy by employing their hybrid method. This category-based blocking reduced their dataset to single category entity clustering problem.

Tejada and colleagues (2002) have proposed an object identification system which can automatically learn the appropriate weights of specific domains for the transformation task by the help of minimized user input. They have utilized specific features of the each domain (e.g., URL of the company domain, phone of

the restaurant domain, code of the airport domain, and so on). However, their proposed system is only applicable to specific categories such as only restaurant category. Thus, their system is not applicable to a dataset that contains restaurants, hospitals, airports, and schools at the same time.

Sarawagi and Bhamidipaty (2002) have proposed an interactive deduplication system that uses active learning. The system employs a trained classifier to classify records as duplicate or not. Their main contribution is, by utilizing active learning, picking up the most influential records to manually classify and obtain high accuracy with the minimized user input. However, their proposed system is not applicable to our record linkage problem since we have an infinite number of different categories.

Cohen and Richman (2002) have presented techniques for entity name matching and entity name clustering tasks that are scalable and adaptive. They define entity name matching task as pairing two different lists from two different data sources and entity name clustering task as clustering coherent entities in a single given list. However, their system is highly different from this study because they employ classification learners for their tasks and provide the number of desired clusters. Moreover, they do clustering or record linkage only on a single category of entities.

Pasula and colleagues (2002) have proposed a model that utilizes Markov Chain Monte Carlo algorithm and Bayesian networks to solve identity uncertainty problem. Since Markov Chain Monte Carlo algorithm is hardly scalable, they also employ distance metrics to fragment dataset into canopies. However, their system is not applicable to our research, because they use external sources for training, and they cluster only single category of entities which are citations.

Grannis and colleagues (2003) have proposed a system that utilizes Felligi-Sunter probabilistic linkage method and Expectation Maximization algorithm. They have used Expectation Maximization algorithm to automatically determine the thresholds of the Felligi-Sunter linkage method. However, their system is not

applicable to our problem since we have multiple category entities, and it is not possible to exploit field weights like they do.

Hotho and colleagues (2003) have proposed an ontology-based Text Documents Clustering Improvement System which utilizes WordNet to add the semantically higher concept words to the document vectors to improve the relevancy of the related documents. For example, if a document contains “*beef*” word and another document contains “*pork*” word, they can be related by adding a higher concept word such as “*meat*”. They have employed Bisecting K-means clustering algorithm for the clustering task. They have obtained best 51.4% F-measure on a subset of Reuters News dataset and 64.6% F-measure on a relatively small sized Java learning lesson related dataset. Their work is highly different from this study because of several reasons:

- i. Their algorithm requires the number of clusters.
- ii. Their documents sizes are extremely larger than the document sizes of our study.
- iii. Instead of clustering the related records, in our study, the same products are tried to be clustered.
- iv. Their methodology is not dynamic. Thus, it is not applicable in E-commerce area where there is a constant arrival of new data.

Fung and colleagues (2003) have proposed a hierarchical clustering system that utilizes frequent itemsets. Their method first generates initial clusters based on frequent global itemsets and then they disjoint them based on inter-cluster similarity to generate final clusters by employing manually pre-defined thresholds. Since their system primarily depends on initial clustering methodology, it is static. Their approach is not applicable to our research because: (i) frequent itemset is not practical for product name clustering since only product title is used, (ii) product name clustering has to be dynamic since new data continuously arrives. Moreover,

their system additionally focuses on categorical clustering (e.g., one cluster is the parent of another cluster) which is not aim of this thesis study.

Jin and colleagues (2003) have proposed a record linkage system which maps data pairs into high-dimensional Euclidean space by using the Euclidean Distance between pairs as the first step and, employs a similarity join algorithm to determine which records are similar or not as the second step. For mapping into high-dimensional space they have used slightly modified version of the FastMap algorithm (Faloutsos & Lin, 1995) and for the similarity-joining operation, they have used the algorithm proposed by Hjaltason and Samet (1998). Their system is greatly different from our proposed system because their proposed approaches are only applicable to the datasets having only single category of entities.

Bhattacharya and Getoor (2004) have proposed an iterative record linkage system that utilizes linkage between the similar objects. For example, if entity *A* and entity *B* are married to entity *C*, then *A* and *B* may be referring to the same person. Their motivation was author resolution rather than paper resolution in the citation domain. They have utilized coauthorship as a linkage criteria. They have experimented on a synthetic dataset along with manually determined varying thresholds. However, their method is not applicable to our research since we do not have structured attributes to utilize.

Cui and colleagues (2005) have proposed a hybrid clustering technique that applies both Particle Swarm Optimization and K-Means clustering algorithms. Since Particle Swarm Optimization algorithm is computationally very expensive, they use a hybrid model. Their systematic approach can be summarized as follows: first globally optimal initial cluster seeds can be found by using the Particle Swarm Optimization algorithm, and then standard K-Means algorithm is employed to obtain better results by using these globally found cluster seeds. The success of the K-Means algorithm highly depends on the initial seed selection. Thus, this method improves the overall performance. Their method is not applicable to our problem

since they have to provide the number of the clusters as an input which is unpredictable in our case.

Bilenko and colleagues (2005) have proposed an adaptive record linkage system that utilizes an online learning approach to identify the same products from different E-commerce websites. They have used the Averaged Perceptron (a space-efficient variation of the Voted Perceptron (Collins, 2002)) algorithm along with multiple similarity functions for each identifying field such as price attribute of digital products or author attribute of books. To obtain the training data, they have exploited UPCs (Universal Product Codes) of the products. Not all products have this attribute, but the products that have are sufficient for the classification algorithms. First the similarity functions are trained, then the HAC algorithm is used to cluster the dataset. Even though their work looks similar to our research, there are several fundamental differences:

- (i) They exploit UPCs for supervised training which is not available in Turkey or many other countries.
- (ii) Even though they mention their method can be applied to continuously arriving data, they do not experiment with such data to cluster and explain details how can it be applied.
- (iii) They heavily use the structured data given in the comparison shopping site where they have obtained their dataset which is not possible in our research.
- (iv) Even though they have experimented only on three different categories of products and have used all these leverages, they have obtained about 40.0% F-measure while we have achieved 96.25% for the much larger number of categories of products.

Shen and colleagues (2005) have proposed an entity matching system based on a generative model that utilizes hard and soft constraints. Performance of

the system is highly affected by the features of the dataset which are the core problem of entity matching systems. For example in the domain of this thesis study, either such features are missing, erroneous or very hard to obtain. Thus, our proposed product code determination system is crucial for product matching domain which we explain in Section 6.1.4.

Bilenko and colleagues (2006) have proposed an adaptive blocking framework to reduce the number of pair wise comparisons. In our study, we propose domain specific blocking methodologies that improve the success rate significantly.

Sahoo and Callan (2006) have proposed the first Incremental Hierarchical Clustering of Text Documents. However, their system and approaches are greatly different from this thesis study. They have experimented on news data, and characteristics of our study and news data domain are completely different. Additionally, they knew the number of categories, and they ordered the input by the date of the news. The order of the input data is extremely important when doing an Incremental Hierarchical clustering. Their tokenization and matching functions are also different than this study. As mentioned earlier, record linkage problem requires highly domain specific solutions.

Yan and colleagues (2007) have proposed adaptive SNMs (Sorted Neighborhood Methods) for more efficient record linkage task. They have especially focused on fixed sliding window size of the SNM method. An SNM sorts the dataset by pre-determined attributes. Thus, sorting based methodologies are not suitable for our research since our aim is to cluster different categories of entities, and each category can have very different attribute schemes. For example, the following two records are the same: “*PS3 DRAGON AGE INQUISITION*”, “*ARAL DRAGON AGE INQUISITION PS3 OYUN*” and if a name attribute based SNM is employed for blocking, they may never get on the same block.

Shen and colleagues (2007) have proposed a source aware approach for entity matching problem that utilizes properties of the source for employing a strict

and relaxed matching function. Their principle is that they assume that there is a best accuracy obtaining strict matching function $S1$. The accuracy of this strict function is very high but recall is limited. They also assume that another function $S2$ can be defined based on properties of the domain which is more relaxed. They propose that by combining these two functions, higher recall can be obtained with minimized loss of precision. Their system requires domain expert to define both $S1$ and $S2$ functions with addition to deciding either the domain is relaxable or not. They only have experimented on entity matching area of mentions.

Inan and colleagues (2008) have proposed a hybrid approach that combines cryptographic techniques and anonymization methods to provide a three-dimensional tradeoff between privacy, cost, and accuracy. Their basic idea is using blocking on an anonymized dataset to reduce the cost of cryptographic methods.

Kim and Lee (2010) have proposed a record linkage blocking system that is based on an iterative Locality-Sensitive Hashing technique that reduces record linkage run time when compared to basic Locality-Sensitive Hashing and Multi-Probe Locality-Sensitive Hashing algorithms. They have called their system as HARRA (HAshed RecoRd linkAge).

Rastogi and colleagues (2011) have proposed a framework for collective highly scalable entity matching task. They have split the dataset into subsets and run multiple instances of matchers on these subsets simultaneously. After each round, they use simple message passing and maximal message passing systems to communicate between each matchers. They continue to run matchers until all matchers are converged. Their entity matching domain has been “author referencing”, and they have experimented on both real and synthetic datasets.

Kannan and colleagues (2011b) have proposed a system that utilizes product images to reinforce text-based classification task of product matching problem. Their proposed Confusion Driven Probabilistic Fusion++ algorithm improve the accuracy of product matching task when compared to the baseline text classifier.

Böhm and colleagues (2012) have proposed a system that does entity matching on a very large scale RDF based (Resource Description Framework) web data. They have composed an entity graph by utilizing RDF format, reflexivity, symmetry, transitivity, and unique mapping per data source restrictions to maximize similarity. By using MapReduce-based assignment approach, they have achieved cluster based scaling. Their system is highly different from our system because they heavily exploit RDF format to calculate similarity and decide which entities are the same. The main focus of their system is having more scalability as they try to make entity resolution on billions of triples (subject–predicate–object in RDF terminology).

Gokhale and colleagues (2014) have proposed a system which utilizes crowdsourcing and does not require any developer interaction to solve entity matching problem. The system uses crowdsourcing to label records and by utilizing this labeling, both the blocking and the matching systems are trained by using Random Forest (a random forest is a set of decision trees (Hastie, et al., 2009)) algorithm. They also have proposed accuracy estimator and difficult pairs locator system. Their system uses crowdsourcing to operate and does not require any developer interaction such as defining rules, writing code, etc. The accuracy of their system highly depends on the correct labeling made by the crowdsourcing. Additionally, the most important part of the system is blocking, specifically, rule generation. Their rule generation system completely depends on the input features, and they do not give any details about how they have generated these features. They say that they use pre-supplied feature library. However, no details are provided about this library. Furthermore, for an efficient rule generation, the supplier has to provide rich and correct features which are extremely hard to obtain in product matching domain. Moreover, they only have evaluated a single category of products matching from two different sources which leave the question unanswered: how the system in a real life scenario (multiple sources that contain multiple categories in an unstructured way) would perform.

Zhang and colleagues (2015) have proposed a system that provides optimizations to max-weight graph matching on multi-partite graphs for multi-source entity matching problem on very large datasets. Their methodologies and aims are highly different from this study.

2.4.2. Related Work about Product Matching Record Linkage

Kannan and colleagues (2011a) have proposed a system for product matching task that is based on linking new product offers to an existing structured repository of products. Their core approach is, first filtering the new product offers by their category via classification. Thus, decreasing the candidate matching pool, and then, scoring them against the existing repository of products and assigning new products to the most score obtained existing products. In the first phase, their matching system which utilizes Binary Logistic Regression classification does offline training to assign scores of new product offers against the products in the repository. This matching system heavily utilizes features of products. It differently penalizes the missing and incorrect features when calculating the score. Once both categorization and matching classifiers are trained, their system is ready to do entity matching. After the system is ready, when a new product offer is arrived, it is classified into one of the categories that are trained. In the second phase, the new product offer is parsed and tagged with its features. After the parsing process, each one of the attributes of the new product is weighted, and the possible candidate matching pool is reduced further by applying an attributes filter. Moreover, finally, their offline trained matching system assigns a similarity score to each one of the existing products in the candidate matching pool, and then the new product is matched to the candidate that gets the highest score. However, this system has restrictions. First of all, it is very hard to obtain such structured product offers repository. They have used Microsoft's Bing Shopping catalog which has more than 5 million product offers. Having such a big and well-structured repository is not doable for every developer. Secondly, relying on such repository requires this

repository to be up-to-date. When a new product is released into the market, probably it will not exist in this repository, and thus, it will get matched into an incorrect item. Their system also relies on the identification of the attributes correctly. If such products do not have any defining particular attributes, such as “*Goody Etli 15 kg Yetişkin Kuru Kedi Maması*”, their system may perform worse when compared to the highly detailed attributes having products such as “*Lenovo IP100 80QQ009ETX Intel Core i5-5200U 2.2GHz / 2.7GHz 4GB 500GB 2GB 920M 15.6 FreeDos Notebook*”. They also have reported better precision and recall for high-value products such as laptops when compared to low-value products such as accessories of smartphones. However, in our system, we do not rely on the attributes definition as heavily as they do. Moreover, attributes may be missing too. Additionally, we do not require such training set and well-structured categories to filter. Thus, we propose that our system is more applicable to a broader set of categories.

Köpcke and colleagues (2012) have proposed a system for better product matching which utilizes product code determination, product category filtering and manufacturer dictionary. However, some of these prerequisites are hard to obtain in real life scenarios. For example, they do not explain how they have obtained exactly that manufacturer dictionary or cleaned product categories. They explain that for evaluating purposes, they have manually cleaned the product categories and the dictionary of manufacturers which is quite not possible when there are hundreds of different E-commerce websites which have thousands of different categories. They probably have been able to fix these manually because they have had obtained their data already from a structured E-commerce product portal unlike us where we use real E-commerce websites. To prove how hard it can be when multiple different E-commerce websites are used, we give an example below where categories of a single product from different E-commerce websites are listed:

- Tablet Aksesuar > Diğer
- Bilgisayar > Bilgisayar Aksesuarlar > Çantalar
- Aksesuar Ürünleri > Çantalar ve Kılıflar > Disk Kılıfları > ADDISON
- Aksesuar > Çanta - Kılıf > Addison HDD-130 Hard Disk Kılıfı Mavi
- Bilgisayar > Veri Depolama > Taşınabilir Disk Aksesuarları > Kılıflar
- Aksesuar & Sarf Malz. > Çantalar > HDD Çantaları
- Bilgisayar > Aksesuarlar > HDD Kılıflar

As can be seen from the above example, it is quite impossible to automatically clean them and make them refer to the same category. Additionally, their used category normalization and product category cleaning depend on batch clustering. When an incremental clustering is being made, they may not work as well as batch clustering. Moreover, they also utilize training dataset. The crucial part of training dataset utilization in product matching task is, a significant percent (at least 25% according to Köpcke and colleagues (2010b)) of the entire dataset need to be provided as the training dataset. When there are hundreds of thousands of products, it is quite not feasible to obtain that amount of training data for that many products. Their code extraction system also mainly depends on the search engine querying which is heavily limited by all of the major search engines right now. Thus, it is not doable when there are too many products. They have used an SVM as a learner and also exploited category based matcher training which favors them greatly. Even though they have used manually fixed product category assignment and search engine exploiting code extraction system, they have obtained best ~66% overall F-measure. When they cluster only mobile phones, they obtain their best F-measure which is 89%. However, as we have mentioned, obtaining clean manufacturer dictionary, directory of each product record and product code extraction via utilizing search engines are not practical. Also in many categories of products, manufacturer name or product codes such as: “*Merkezi Kilit Motoru Tek Motor 2 Kablo 12 VOLT (Arka - Bagaj)*” do not exist. On our entire

dataset without such strict dependencies, we obtain 96.25% macro-average F-measure. Additionally, our algorithm is incremental which is a must in record linkage of products.

Petrovski and colleagues (2014) have proposed a system that utilizes product feature extraction to achieve higher accuracy product matching. Their system mainly composed of feature extraction methodologies that use manually defined rules. Moreover, they utilize automatic linkage between free text properties and structured product properties to obtain features. By using a combination of training on the structured data, manually defined regular expressions, and the dictionary search they have obtained the following properties of the products: brand, model, storage, display, processor and dimensions. With using all these properties of the products, they have employed Silk Link Discovery Framework (Isele & Bizer, 2011) that is based on genetic programming approach and have obtained 82.4% F-measure at best. However, even though their domain is similar to ours, there are fundamental differences. First of all, they only have experimented with digital products category by employing the Naive Bayes classifier to classify their dataset. With this high limitation, they have heavily exploited the unique characteristics of the specific category. In our case, we do not filter our domain to a certain category and exploit its characteristic features. Because in Turkey, we do not have such structured hierarchical category having website (they have used Amazon which sells products almost from every product category) for training. Additionally, they have heavily employed training and manually defined rules to obtain better feature extraction which requires expensive human laboring in a big dataset especially when there are multiple categories. Furthermore, many different categories of products do not contain such product features to utilize. Thus, it may not be possible to obtain higher accuracies. Their published F-measure is also based on 2500 matching pairs and 2500 non-matching pairs rather than clustering a certain number of different products such as in our case. We have clustered 250

different products that have 1721 records. This matching versus non-matching classification method also simplifies the task when compared to our clustering task.

2.4.3. Surveys about Record Linkage

There are several surveys published on record linkage problem recently and they can be checked to have a more detailed idea. Koudas and colleagues (2006) have made a very extensive report about similarity metrics and algorithms that are used in clustering to solve record linkage problem. They have discussed data quality, linkage methodology, learning core measures, linkage based measures, efficient algorithms for approximate join and clustering/partitioning algorithms. Winkler (2006) has published an extensive research report about record linkage problem. They have covered standardization, string comparators, parameter estimation and link analysis topics. Hall and Fienberg (2010) have presented a survey of privacy-preserving record linkage methodologies and algorithms. They have pointed out two main challenges of the privacy-preserving record linkage: since the privacy is a concern, it is difficult to find out optimal parameters and development of techniques is hard due to uncertainty in the dataset. Köpcke and Rahm (2010a) have published an extensive overview study about frameworks that are used in entity matching area. They have analyzed 11 proposed frameworks for entity matching with considering blocking support, uses training data or not and how they have been evaluated. They have concluded that the recent frameworks are advancing toward to training data utilizing systems. Additionally, another very important observation they have made is that all of the frameworks focus on offline matching. From these points of views, our study is different from many of the recent studies. They also have stated that due to the different methodologies, measures, and test problems it is not possible to determine effectiveness and efficiency of the systems comparatively. They did not conduct experiments themselves for more equitable comparison.

Köpcke and colleagues (2010b) have presented a comprehensive report on the evaluation of more than 20 entity resolution approaches on real-world datasets. They have experimented on bibliographic and E-commerce domains by using four different datasets that each one is composed of two different data sources. They have used FEVER (Köpcke, et al., 2009) framework for parameter tuning and more systematic evaluation of different approaches. They have experimented on both learning using and not using frameworks and compared all of the frameworks. Their research on E-commerce domain is similar to ours. However, there are several fundamental differences. They have used datasets only from two different data sources when experimenting while we have experimented on combination of 50 different data sources. Their data sources were providing better structured data which made their task easier. They have not experimented with any incremental approach, and they have chosen products only from manually determined categories of electronic products. Even though there are critical differences, their work sheds light on a very important conclusion about our research domain (E-commerce). While the tested approaches can obtain very high F-measures on the bibliographic datasets, success of the very same approaches severely decrease on E-commerce domain. For example, while FEBRL (Christen, 2008) FellegiSunter (Fellegi & Sunter, 1969) obtains best 97.4% F-measure on bibliographic datasets, its maximum F-measure for the E-commerce domain is only 53.8%. Among all of the approaches, they have obtained best 97.6% F-measure for the bibliographic datasets and 70.7% F-measure for the E-commerce domain. One another very important deduction of their research is, while learning based approaches can obtain higher success rates by using multiple attributes, they significantly increase the run time and become infeasible on large datasets. Additionally, in E-commerce domain, it requires a very significant proportion of the dataset to be labeled for training to obtain decent results. For instance, even when thousands of records are used for training, the performance increase is minimal or even negative. By using learning based approaches, maximum F-measure of the E-commerce dataset

increases from 70.7% to 71.3% and the maximum F-measure of the second dataset decreases from 62.2% to 60.1% according to their experiments.

Aggarwal and Zhai (2012) have published an extensive survey of text clustering algorithms. Christen (2012) has published an extensive review of indexing techniques that are used in record linkage problem and has provided comprehensive experimental comparison results. Indexing techniques are used to reduce the number of candidate record matching. Thus, the record-linkage performance is improved via fewer computations. Moreover, with lesser mismatch comparisons better matching quality is obtained. Their survey is about Traditional Blocking, Sorted Neighbourhood Indexing, Q-gram Based Indexing, Suffix Array Based Indexing, Canopy Clustering and String-map Based Indexing by using manually assigned parameters and thresholds. In this study, we have used Traditional Blocking and Q-gram Based Indexing (when character based N-Gram is applied Traditional Blocking becomes Q-gram Based Indexing) with each record can be assigned to multiple blocks because we do not want to miss any true candidates. Also, according to the tests they have conducted, Q-gram Based Indexing and Traditional Blocking usually perform better than the other techniques.



3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

3. ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING

In this section, the algorithms and metrics used in this thesis study are presented thoroughly.

3.1. Standardization/Preprocessing Methods

Standardization is also called data cleaning or attribute-level reconciliation (Gu, et al., 2003). Standardization prevents possible mismatch of true matches by correcting different expression of the same attributes (Winkler, 1995).

The applied techniques are generally removing stop-words, upper case/lower case conversion, spelling variations correction, standardization of the representation of various attributes (e.g., kg/pound conversion), removing non-meaningful words, characters and signs.

3.1.1. Spelling Correction

Spelling correction is the task of the finding most suitable word for a given word w in the given dictionary D (Toutanova & Moore, 2002). Spelling correction can be useful when the contextual analyses are made such as in sentiment analysis. It is rather common that user generated content such as product comments contain spelling mistakes. For example word “*home*” can be incorrectly spelled as “*homm*” and spelling correction process can fix it.

3.1.2. Diacritics Transformation

Diacritics are marks that are added to the letters for differentiation of how they are pronounced (Wells, 2000). In this study, when we refer to Diacritics Transformation, we mean that converting diacritics having letters into their Latin forms. For example Turkish letter “*ş*” is converted into letter “*s*”.

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

3.1.3. Lower Case Conversion

Lower case conversion is a very easy yet, very fundamental process in text processing. In many cases, both in record linkage and sentiment analysis area, strings are compared and string comparison task is case sensitive in many of the programming languages including the C# programming language. Thus, converting all letters into their lower forms ensures same words with different cases are treated as the same. For example “*Mouse Set*” becomes “*mouse set*”.

3.1.4. Data Transformation

Data transformation is the task of standardization of the representation of various attributes. For example converting “*kg*” into “*pound*” or converting “*”*” into “*inc*”. This process is the most important in the record linkage problem since different E-commerce sites tend to define the same attributes in different formats.

3.1.5. Punctuation and Non-ASCII Character Removal

The task of character removal changes according to the context. In this study, we have removed punctuation characters from the documents as well as several other characters specific to the domain. For example for the record linkage task, the character “(” is replaced with the space character while for the sentiment analysis task, it is removed from the given string. We also remove non-ASCII characters after diacritics transformation is applied.

3.2. Tokenization and Bag-of-Words Presentation

Bag-of-Words is the task of splitting text into words by space character where each word represents a feature (Sriram, 2010) in NLP area. In BOW, the order of the words are not important. Tokenization is the task of converting records into a meaningful feature set by some method. In NLP, tokenization is usually applied by splitting text by the space character into a word set like in BOW. However, for example, if word level N-gram transformation is applied for

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

tokenization, not every word becomes a feature but a cluster of words become a feature. Usually, both BOW and tokenization tasks are used as the same task in many of the literature studies in NLP area according to our observations. But tokenization process alone is not enough because many of string related algorithms such as distance or similarity algorithms use numerical weights rather than strings. Thus, features of the records still have to be weighted into numerical values. Then they can be represented as vectors. For example “samsung galaxy s4” and “samsung note 4” documents can be represented as shown in Table 3.1 by weighting them with frequency of the words.

Table 3.1. Vectoral representation example

Doc ID	Words					Vectoral Representation
	samsung	galaxy	s4	note	4	
1	1	1	1	0	0	[1, 1, 1, 0, 0]
2	1	0	0	1	1	[1, 0, 0, 1, 1]

When doing vectoral computations, according to our experiments, the direction of the vectors can be more important than their magnitude. In order to remove effect of the magnitude from the vectors, vector normalizations are applied. In this study, L_p space vector normalizations are employed. The formula of L_p normalization is presented below where x_i is the i th element of the vector x , n is the number of the elements in the vector, and p is an integer from 1 to 5:

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

3.3. N-Grams

An N-gram is an N-character slice of a longer string which is applied continuously (Cavnar & Trenkle, 1994). For example 2-gram representation of term “school” is “sc”, “ch”, “ho”, “oo”, “ol”. N-character slicing can be replaced with N-words slicing as well. For instance the 2-gram word representation of “we love programming” is “we love”, “love programming”.

3.4. Weighting Schemes

The employed weighting scheme determines how the document is vectorized. Majority of the string based algorithms require numerical inputs such as Manhattan Distance. Moreover, SVMs also work only with numerical input. Because of these reasons, each document has to be tokenized then each token (i.e., feature) is assigned a numerical weight value and it is referred as representation of documents in Vector Space Model (Reed, et al., 2006). When documents are represented in Vector Space Model, each term is assigned a numerical value that which represents importance of that term in the given document (Salton & Buckley, 1988). According to our experiments, how the terms are weighted directly affects the performance of both classification and clustering tasks. Thus, we have experimented with different weighting schemes for both record linkage and sentiment analysis tasks. To assign weights to features, the most commonly used values are term frequency (TF), inverse document frequency (IDF), and relevance frequency (RF) of the tokens in the documents (datasets).

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

3.4.1. Term Frequency Variants

Variants of TF weights (Paltoglou & Thelwall, 2010) are shown below where $f_{t,d}$ is the term frequency of term t in document d , k_1 and b are constants of BM25, dl is document length and avg_dl is the average document length of all documents.

Table 3.2. Variants of TF weighting

Name	Formula
Binary:	$= \begin{cases} 0 & \text{if term } t \text{ is not in document } d \\ 1 & \text{otherwise} \end{cases}$
Raw Frequency:	$= f_{t,d} = \text{frequency of term } t \text{ in document } d$
Log Normalization:	$= 1 + \log(f_{t,d})$
Double Normalization:	$= 0.5 + 0.5 \times \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
BM25:	$= \frac{(k_1 + 1) \times f_{t,d}}{k_1 \times \left((1 - b) + b \times \frac{dl}{avg_dl} \right)}$

3.4.2. Inverse Document Frequency Variants

Variants of IDF (Paltoglou & Thelwall, 2010) weights are shown in Table 3.3 where N is the number of documents, N_t is the number of documents that contain term t , N_p is the number of positive class documents, N_{pt} is number of positive class documents that contain term t , N_N is number of negative class documents, and N_{nt} is number of negative class documents that contain term t .

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

Table 3.3. Variants of IDF weighting

Name	Formula
Unary IDF:	$= 1$
IDF:	$= \log \frac{N}{N_t}$
IDF Smooth:	$= \log \left(1 + \frac{N}{N_t} \right)$
IDF Max:	$= \log \left(1 + \frac{\max_{\{t' \in d\}} N_{t'}}{N_t} \right)$
Probabilistic IDF:	$= \begin{cases} 0 & \text{if } N = N_t \\ \log \frac{N - N_t}{N_t} & \text{otherwise} \end{cases}$
Delta IDF:	$= \log \left(\frac{N_P \times N_{nt}}{N_N \times N_{pt}} \right)$
Delta BM25 IDF:	$= \log \left(\frac{(N_P - N_{pt} + 0.5) \times N_{nt} + 0.5}{(N_N - N_{nt} + 0.5) \times N_{pt} + 0.5} \right)$

3.4.3. Relevance Frequency

RF is a new metric proposed by Lan and colleagues (2009) and claimed to be more successful than other known techniques for text categorization task. The formula of RF is shown below:

$$\text{RF IDF} = \log \left(2 + \frac{N_{pt}}{\max(1, N_{nt})} \right)$$

3.5. Distance Metrics

The employed distance metrics in the study are presented in Table 3.4.

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

Table 3.4. Distance Metrics

Algorithm	Formula	Range
Euclidean Distance: Where q and p are feature weight vectors.	$d(q, p) = \sqrt{\left(\sum_{i=1}^n (q_i - p_i)^2\right)}$	$0 - \infty$
Squared Euclidean Distance: Where q and p are feature weight vectors.	$d(q, p) = \left(\sum_{i=1}^n (q_i - p_i)^2\right)$	$0 - \infty$
Manhattan Distance (Taxicab Geometry): Where q and p are feature weight vectors.	$d(q, p) = \left(\sum_{i=1}^n q_i - p_i \right)$	$0 - \infty$
Maximum Distance (Chebyshev Distance): Where q and p are feature weight vectors.	$d(q, p) = \max_i(q_i - p_i)$	$0 - \infty$

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

Table 3.4. Distance Metrics (Continued)

Jaccard Distance: Where A and B are string vectors. This algorithm is same as <i>Tanimoto Binary Distance</i>.	$d_J(A, B) = 1 - s_J(A, B)$ $= 1 - \frac{ A \cup B - A \cap B }{ A \cup B }$	0 – 1
Pearson Correlation Distance: Where x and y are data objects, and N is the total number of attributes (Philip, 2010).	$Pearson(x, y) = \frac{\sum xy - \frac{\sum x \sum y}{N}}{\sqrt{(\sum x^2 - \frac{(\sum x)^2}{N})(\sum y^2 - \frac{(\sum y)^2}{N})}}$	0 – 2
Average Kullback Leibler Divergence Distance: Where a and b are feature weight vectors and P is probability function (Sutton, 2007).	$\bar{d}(a, b) = \frac{\left(\sum_{i=1}^n (a_i) \times \log_2 P(a_i) + \sum_{i=1}^n (b_i) \times \log_2 P(b_i) \right)}{2 \times \log_2 2}$	0 – ∞

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

Table 3.4. Distance Metrics (Continued)

Cosine Distance: Where q and p are feature weight vectors	$d_c(q, p) = 1 - s_c(q, p) = 1 - \frac{\sum_{i=1}^n q_i p_i}{\sqrt{\sum_{i=1}^n q_i^2} \sqrt{\sum_{i=1}^n p_i^2}}$	0 – 1
Bray Curtis Distance: Where q and p are feature weight vectors (Souza, 2016).	$d(q, p) = \frac{\sum_{i=1}^n q_i - p_i }{\sum_{i=1}^n q_i + p_i }$	0 – 1
Canberra Distance: Where q and p are feature weight vectors.	$d(q, p) = \sum_i \frac{ q_i - p_i }{ q_i + p_i }$	0 – ∞
Normalized Canberra Distance: Where q and p are feature weight vectors.	$d(q, p) = \frac{1}{n} \sum_i \frac{ q_i - p_i }{ q_i + p_i }$	0 – 1

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

Table 3.4. Distance Metrics (Continued)

Hamming Distance: Where i and j are objects, and k is the index of the respective variable reading y out of the total number of variables n (Schulz, 2008).	$d^{HAD}(i, j) = \sum_{k=0}^{n-1} [y_{i,k} \neq y_{j,k}]$	$0 - \infty$
Dice Distance: Where A and B are strings.	$d_D(A, B) = 1 - s_D(A, B)$ $= 1 - \frac{2 A \cap B }{ A + B }$	$0 - 1$
Tanimoto Non-Binary Distance: Where A and B are feature weight vectors. This algorithm is same as <i>Extended Jaccard Distance</i> (Karlsson, 2012).	$d_T(A, B) = 1 - s_T(A, B)$ $= 1 - \frac{\sum_i (A_i B_i)}{\sum_i (A_i^2) + \sum_i (B_i^2) - \sum_i (A_i B_i)}$	$0 - 1$

3.6. Classification

In this study, classification task is used to classify product reviews into negative and positive classes. For this task, only a Support Vector Machine (SVM)

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

classifier is used because their success is proven in the literature when only statistical methods are applied. Since SVMs are highly complex algorithms, we have used LibSVM (Chang & Lin, 2011) which is an open source SVM framework. The used version is 3.21 x64. Unfortunately LibSVM does not have native C# support. For integration, we have used open source project LibSVMsharp (Erhan, 2015) as a wrapper.

An SVM is a discriminative classifier formally defined by a separating hyperplane. The algorithm takes labelled training data as an input and outputs an optimal hyperplane which can categorize new examples (Team, 2016). The goal of an SVM is finding hyperplanes that separate the training data with largest margin. Linear SVM examples are illustrated in Figure 3.1 and Figure 3.2.

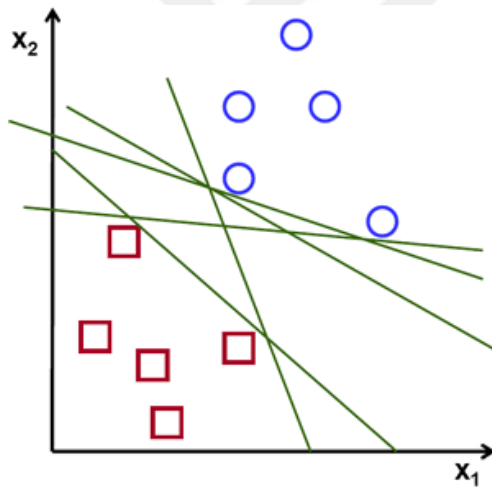


Figure 3.1. Possible example solutions for a Linear SVM (Team, 2016)

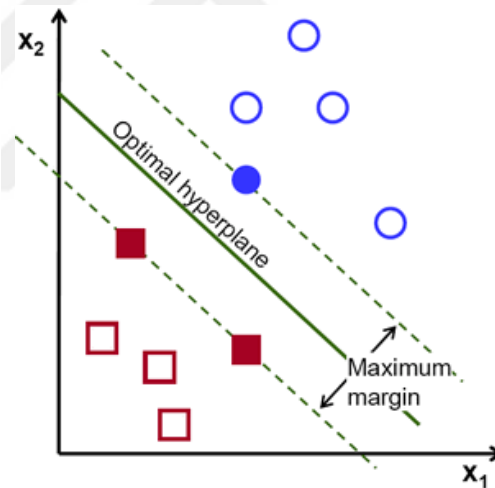


Figure 3.2. Optimal solution for a Linear SVM where solid color filled shapes are support vectors (Team, 2016)

The notation used to define a hyperplane is $f(x) = \beta_0 + \beta^T x$ where β is the *weight vector*, β_0 is the *bias*, and x is a support vector (Team, 2016). The training examples that are closest to the hyperplane are called support vectors.

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification by using kernels which map the inputs into higher-dimensional feature spaces (Hofmann, et al., 2008). Most commonly used kernel methods are Linear, Polynomial, Radial Basis Function, and Sigmoid (Hsu, et al., 2010).

3.7. Clustering

Clustering is the process of grouping similar objects in an unsupervised fashion (Singh & Husain, 2014). In this study, clustering techniques are employed to conduct record linkage on the E-commerce products. Our aim is clustering same E-commerce products which are sold in different E-commerce websites, into the same clusters with highest possible precision and recall.

3.7.1. Searching/Blocking

Searching or blocking is used to reduce the number of comparisons of record pairs by eliminating possibly unrelated records getting paired (Gu, et al., 2003). When doing record linkage on very big datasets, it is not practicable to compare every pair of records. Thus, blocking is extremely important in the record linkage task. It speeds up the processing and also can improve the success rate by preventing miss-matches. In our study, we have employed Traditional Blocking to some degree. However, our true success comes from matching rules which block further comparison of unrelated pairs. Our proposed matching system is explained in Section 6.1.

3.7.2. Hierarchical Agglomerative Clustering (HAC)

Hierarchical clustering aims to construct clusters by recursively partitioning or merging the objects (Rokach & Maimon, 2005). It can be done in two different approaches. The first approach works in a bottom-up fashion and it is also known as AGNES (Agglomerative Nesting) (Kassambara, 2015). When doing

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

HAC, in the beginning each object itself is a singleton cluster and at each turn, the most similar or the nearest ones are merged. The second approach works in top-down fashion and also known as DIANA (Divisive Analysis) (Kassambara, 2015). When doing DIANA, in the beginning all of the objects are inside a single big cluster and at each turn, the most dissimilar or the farthest objects in the bigger clusters are split into sub clusters. In this study, only agglomerative approach is employed since our system does dynamic/incremental clustering. When doing dynamic clustering, you do not have all of the data at once. Thus, the new data can only be merged with the existing clusters or put into a new cluster. An example for AGNES and DIANA is illustrated in Figure 3.3.

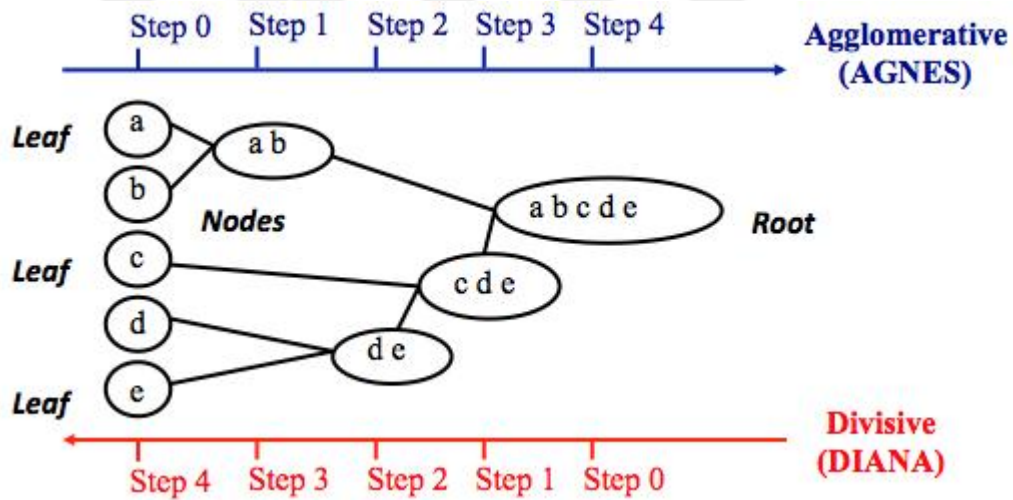


Figure 3.3. An example for Agglomerative and Divisive Hierarchical Clustering (Kassambara, 2015)

The linkage criteria determines how the distance between different observations is calculated. Since each cluster can have multiple instances, we have experimented with multiple different linkage algorithms that are shown in Table 3.5 where A and B are the compared sets and d is the chosen distance metric, to

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

calculate the final distance between cluster pairs. For more information about linkage criteria, we refer to the survey (Murtagh, 1983).

Table 3.5. Clustering Linkage Criteria

Name of the Linkage Criteria	Formula
Maximum or Complete-Linkage Clustering (Defays, 1977)	$\max\{d(a, b) : a \in A, b \in B\}$
Minimum or Single-Linkage Clustering (Sibson, 1973)	$\min\{d(a, b) : a \in A, b \in B\}$
Mean or Average Linkage Clustering, or UPGMA (Sokal & Michener, 1938)	$\frac{1}{ A B } \sum_{a \in A} \sum_{b \in B} d(a, b)$
Centroid Linkage Clustering, or UPGMC (Jain & Dubes, 1988) Where c_S and c_T are the centroids of clusters S and T , respectively	$\ c_S - c_T\ $
Wards Linkage: Where ESS is Error Sum of Squares and X_{ijk} is the value for variable k in observation j belonging to cluster i (Wiesner, 2016).	$d(A, B) = ESS(A \cup B) - (ESS(A) + ESS(B))$ $ESS(A) = \sum_i \sum_j \sum_k x_{ijk} - \bar{x}_{ik} ^2$

A graphical explanation of Single-Linkage, Complete-Linkage, Average Linkage, and Centroid Linkage is presented in Figure 3.4. Examples for Complete-Linkage, Single-Linkage, Average Linkage, and Wards Linkage are illustrated in Figure 3.5, Figure 3.6, Figure 3.7, and Figure 3.8, respectively.

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

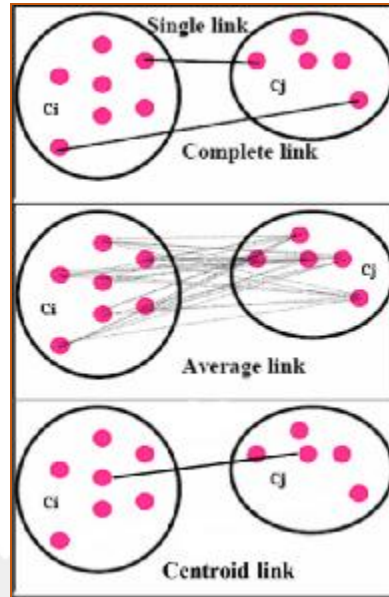


Figure 3.4. Graphical representation of Single-Linkage, Complete-Linkage, Average Linkage, and Centroid Linkage methods (Veress, 2013)

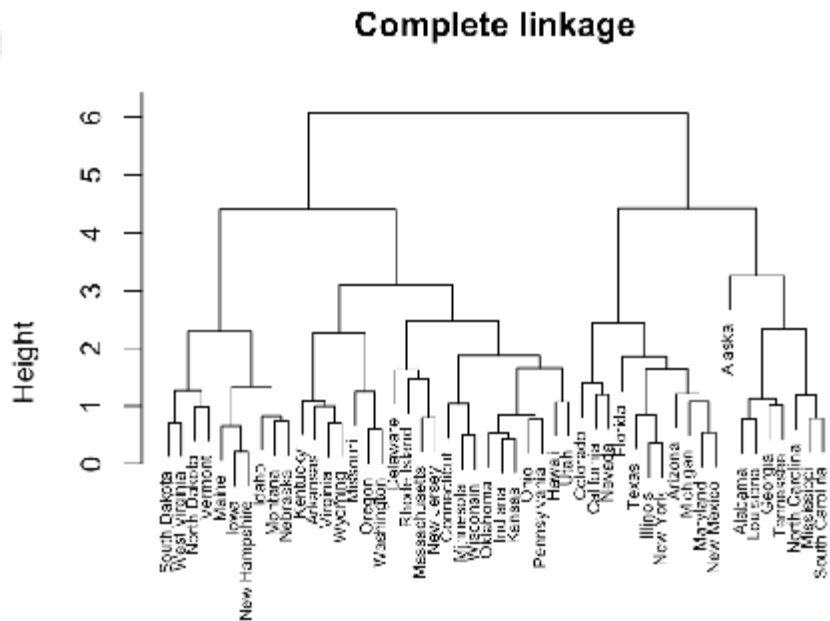


Figure 3.5. Complete-Linkage example (Kassambara, 2015)

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

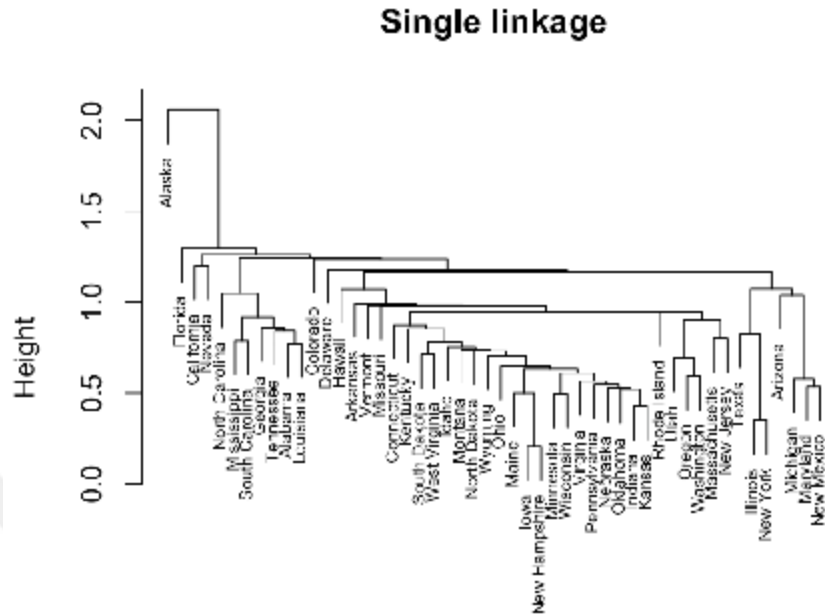


Figure 3.6. Single-Linkage example (Kassambara, 2015)

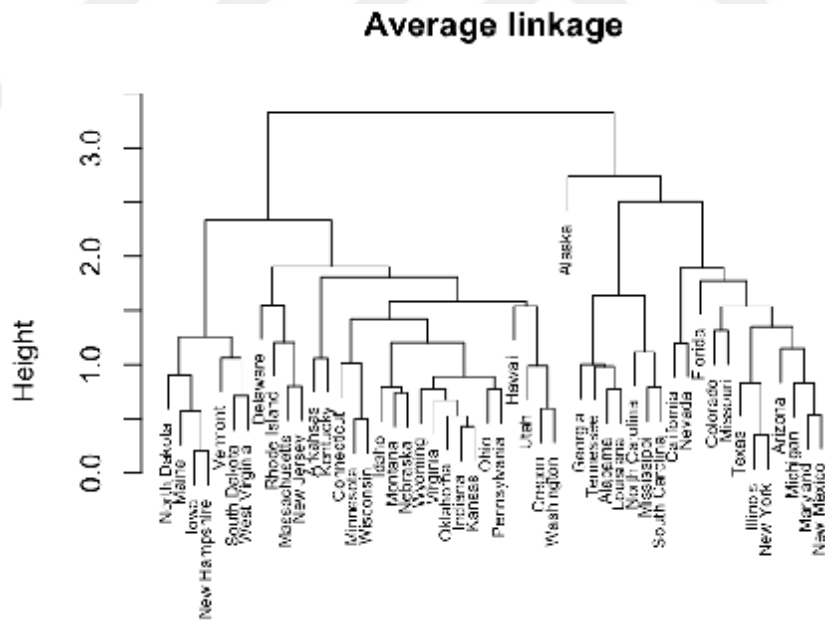


Figure 3.7. Average Linkage example (Kassambara, 2015)

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

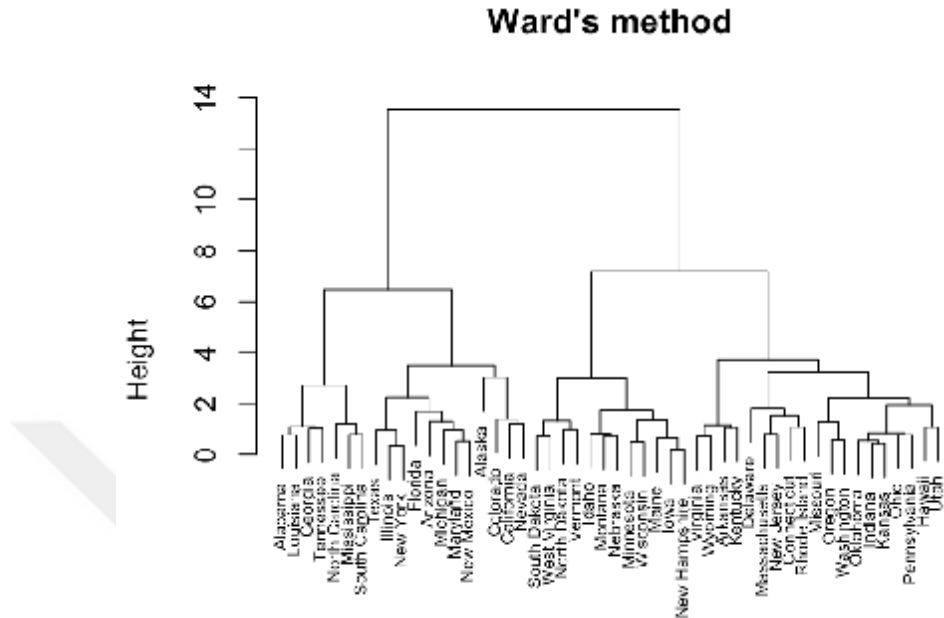


Figure 3.8. Wards Linkage example (Kassambara, 2015)

3.8. Performance Evaluation Methods

Performance evaluation is a critically important aspect of both classification and clustering algorithms to develop a high quality system. The performance measurement of a classification task is straight forward. F-measure metric is the most widely used measure for performance evaluation of classification.

There are two different approaches to evaluate performance of clustering algorithms. The first approach is internal evaluation which uses the clustered data itself. The second approach is external evaluation which uses labelled data to calculate accuracy of the clustering. We have not used any internal performance evaluation metrics since high success rates obtained by using internal evaluation metrics do not necessarily mean to be effective in real-world applications and can be misleading (Manning, et al., 2009). For more information about internal evaluation metrics we refer to (Liu, et al., 2010).

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

When using external evaluation metrics, there are several different implementations of the same metric. The most widely used external evaluation metrics are ARI (Adjusted Rand Index) and F-measure for calculating the performance of the clustering task. But, the problem is, there are multiple ways to calculate F-measure in clustering domain. We have found in the literature at least four different F-measure implementation for evaluating the performance of clustering task. They are Pair counting based F-Measure (Manning, et al., 2009), B-Cubed based F-Measure (Bagga & Baldwin, 1998), Set-Matching based F-Measure (Meilă, 2007), and Editing-distance based F-Measure (Pantel & Lin, 199-206). Since there is no widely accepted performance metric and implementation for clustering task, it is difficult to compare performance rates of different studies especially when the implementation details of the evaluation metric are not presented. Thus, all of the clustering performance measurements we present in this study are calculated by using ELKI (Environment for Developing KDD-Applications Supported by Index-Structures) open source data mining software (Schubert, et al., 2015). Using ELKI for evaluation provides certainty and objective comparability for future studies. ELKI also does prove our claim because ELKI provides five different F-measure evaluation scores that are used in the literature which are based on different measurements.

3.8.1. Precision, Recall, Accuracy and F-measure

In information retrieval area, precision is the ratio of retrieved objects that are relevant while recall is the ratio of related object that are retrieved (Van Rijsbergen, 1979). However, for defining accuracy metric in classification task, a more detailed context is used. In this context, there are four variables employed which are shown below (Powers, 2007):

- TP (True Positives): Correctly predicted as positive instances
- FP (False Positives): Incorrectly predicted as positive instances

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

- TN (True Negatives): Correctly predicted as negative instances
- FN (False Negatives): Incorrectly predicted as negative instances

The accuracy of classification task can be calculated as shown below (Veropoulos, et al., 1999):

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

In classification, Precision, Recall and F-measure are calculated as shown below (Forman, 2003):

$$\begin{aligned} \text{F-measure} &= \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \\ \text{Precision} &= \frac{TP}{TP + FP} \\ \text{Recall} &= \frac{TP}{TP + FN} \end{aligned}$$

As can be seen above definitions, it is not straight to determine TP , TN , FP and FN values in a clustering task as in a classification task.

3.8.2. Adjusted Rand Index

ARI is another very commonly employed metric to evaluate clustering performance. The formula of ARI (Yeung & Ruzzo, 2001) is shown below where X is the c partition generated by the clustering algorithm, Y is the d partition that is correctly labelled, n is the total number of objects ($X + Y$), n_{ij} is the number of

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

objects that are in both class X_i and Y_j , n_i is the number of objects in class X_i , and n_j is the number of objects in class Y_j ;

$$ARI = \frac{\sum_{i,j} \binom{n_{ij}}{2} - \left[\sum_i \left(\binom{\sum_j n_{ij}}{2} \right) \sum_j \left(\binom{\sum_i n_{ij}}{2} \right) \right] / \binom{n}{2}}{\frac{1}{2} \left[\sum_i \binom{n_i}{2} + \sum_j \binom{n_j}{2} \right] - \left[\sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2} \right] / \binom{n}{2}}$$

3.9. Feature Selection

Feature selection is the process of selecting a subset of the feature set mainly for the following reasons:

- Better generalization by reducing overfitting (Bermingham, et al., 2015)
- Shorter training times due to reduced feature count
- Easier interpretation by simplified models (James, et al., 2013)
- Possibly improving classification performance by eliminating noisy features (Jain & Zongker, 1997)

In this study, we have employed feature selection on sentiment analysis task to improve the accuracy and running time of the classification by reducing the feature set size. However, in record linkage area, we have not employed any feature selection algorithm since only title of the E-commerce products are used and they usually do not contain any noisy data. Thus, elimination of the rarely existing noisy data is not doable without elimination of the relevant data. Moreover, feature selection methods usually require batch data preprocessing. Hence, feature selection is not applicable to our proposed record linkage model.

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

3.9.1. Entropy and Information Gain

Entropy is a measure of uncertainty in the result of an event. In classification, it is about how much uncertainty a feature has when classifying instances. For example if a feature exists only in a single class, it has 100% certainty. Therefore, it has 0 entropy. Or if a feature exists one time in class A and one time in class B, it has 0% certainty. Thus, it has 1.0 entropy. The formula of entropy (Quinlan, 1986) is shown below where D is the current set, C is the set of classes in D , $p(x)$ is the ratio of the elements in class x against the number of elements in D :

$$Entropy(D) = H(D) = - \sum_{x \in C} p(x) \log_2 p(x)$$

The Information Gain (IG) of the feature A can be calculated by the calculating difference between the entropies of the set D before and after it is split on attribute A . The formula of IG (Quinlan, 1986) is given below where $H(D)$ is the entropy of set D , T is the subsets composed after splitting set D by attribute A , $p(t)$ is the ratio of elements having value t against the number of elements in set D , $H(t)$ is the entropy of the subset having value t :

$$IG(A, D) = H(D) - \sum_{t \in T} p(t) H(t)$$

From the above equation, it is clear that the attribute which has the highest IG having attribute is the one with the least entropy. Thus, when doing

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

classification, the most IG having attributes from the training set can be chosen by calculating the entropy of the each attribute first, then sorting them in ascending order by their entropy values. With this sorting, it can be said that the top attributes are being the most certainty providing features. Therefore, they can be assumed as the most important attributes. However, IG may have a very big drawback in certain scenarios. For example, if we want to classify a dataset of students, and if the dataset has student ID attribute, it may get the highest IG since each student has a unique ID number. But, student ID attribute is not a good attribute to classify new examples. So, selecting the attribute may cause overfitting of the training set.

In this study, we have calculated entropy of the each feature in the training set and sorted them in ascending order to select top features.

3.9.2. Chi-Square

Chi-Square (CHI2) is used to calculate terms independence to their respected class. The higher they are depended to their class, the higher CHI2 score they have. It is logical to assume that if a feature is highly depended to class then this feature can be more relevant to the context. Thus, they can be more effective in classification task. We refer to study of Manning and colleagues (2009) for more information about how CHI2 is used in feature selection task. The formula of CHI2 (Yang & Pedersen, 1997) algorithm is shown below where t is a term, c is a class, A is the number of times t and c co-occur, B is the number of times t occurs without c , C is the number of times c occurs without t , D is the number of times neither c nor t occurs, and N is the total number of documents:

$$CHI2(t, c) = \frac{N \times (AD - CB)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)}$$

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA

The above equation only calculates CHI2 measure of a single term according to a single class. If the class c and the term t are independent, then CHI2 (t, c) will be equal to 0. Being independent means, that particular term is not related to that particular class. So, that term is not useful for training. Moreover, we need to calculate average CHI2 measure of every term in the training set according to every class in the training set to be able to rank them by their average CHI2 score. So, we can possibly select the most useful features for training the classification model. The formula of how the average CHI2 scores of terms are calculated is shown below (Yang & Pedersen, 1997) where m is the number classes in the training set that contains term t and c_i is the i -th class that contains term t :

$$CHI2_{avg}(t) = \sum_{i=1}^m \frac{1}{m} \times CHI2(t, c_i)$$

3. UTILIZED ALGORITHMS AND METRICS FOR CLASSIFICATION AND CLUSTERING Furkan GÖZÜKARA



4. PROPOSED FOCUSED WEB CRAWLER SYSTEM

This section explains the general architecture of the EcCrawler as well as the challenges encountered during the development of the EcCrawler and possible solutions to these problems.

4.1. The Architecture of EcCrawler

EcCrawler starts by initializing public static variables, functions, and classes. In the second stage, main tasks that are URL Handler, Crawler, Pages Processor, and Global Statistics Handler are started independently from the main thread. Thus better overall application performance and the UI (User Interface) responsiveness³ are obtained. The UI thread and the main thread are the same thread in this study and this is the default setup of WPF (Windows Presentation Foundation) applications in the .NET framework. Figure 4.1 shows the workflow of EcCrawler. In the following subsections, implementation details of the main tasks such as initialization, main thread, URL handler, URL crawler, database communication, pages processor, etc. are presented.

³ Chapter 6 — Using Multiple Threads, <https://msdn.microsoft.com/en-us/library/ff649143.aspx>

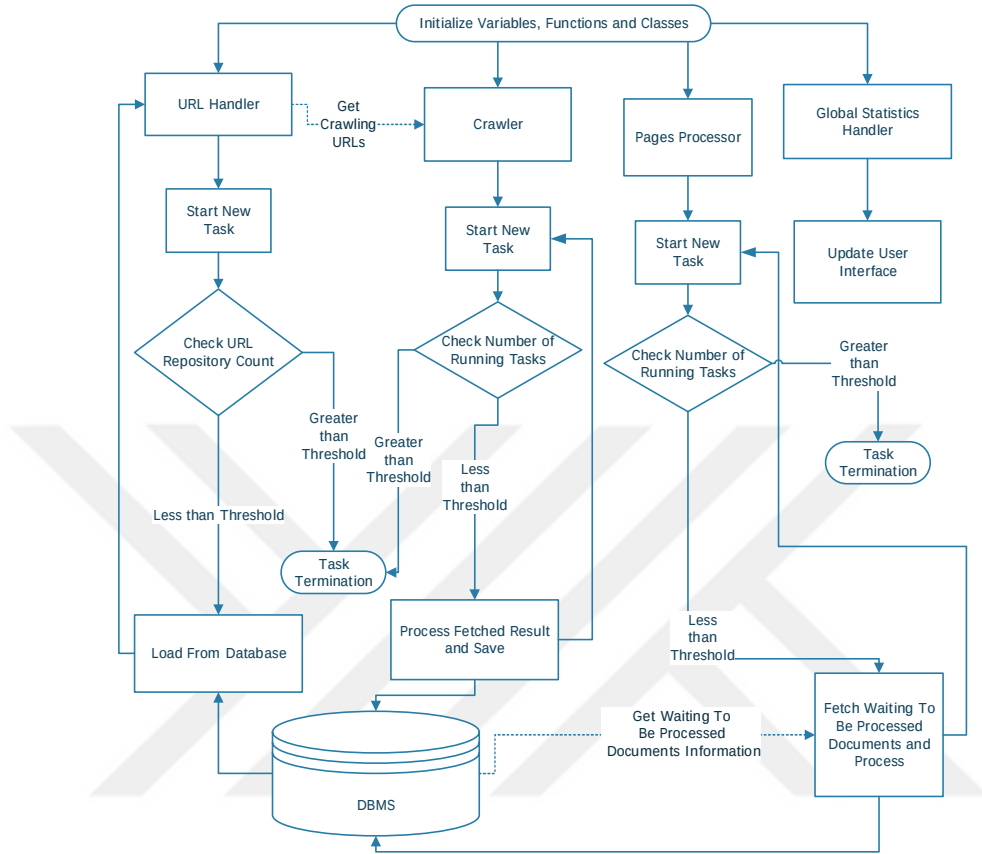


Figure 4.1. Flowchart of EcCrawler

4.2. Database Design of EcCrawler

The tables used in EcCrawler is shown in Figure 4.2 and their purposes are listed below:

- **tblRootSites:** This table is used to store basic information about root sites such as their IDs, filtering rules when crawling, how many concurrent crawling, and so on.
- **tblRootSitesIdentifiers:** This table is used to store wrappers' information for each E-commerce website such as identifier type, HTML object type, HTML object type name, and so on to extract products' information.

4. PROPOSED FOCUSED WEB CRAWLER SYSTEM Furkan GÖZÜKARA

- **tblRootSitesFeaturesIdentifiers:** This table stores wrappers' information for each E-commerce website such as identifier type, HTML object type, HTML object type name, and so on to extract products' features.
- **tblCrawlUrls:** This is the biggest table of the EcCrawler where all information of crawled pages and products are stored.
- **tblProductCategories:** This table is used to hold information about categories discovered from all E-commerce websites.
- **tblProductPagesCategories:** This table is used to save category information of the crawled products.
- **tblWordsProducts:** This table stores words/tokens that are extracted from the title of products to provide faster search of products by using an index made over these words/tokens.
- **tblProductComments:** This table is used to store discovered product comments.
- **tblAutoClusters:** This table is used to store generated clusters by our proposed record linkage system.

More detailed information about the tables are provided in Figure 1 and Figure 2 that are given in the Appendix section.

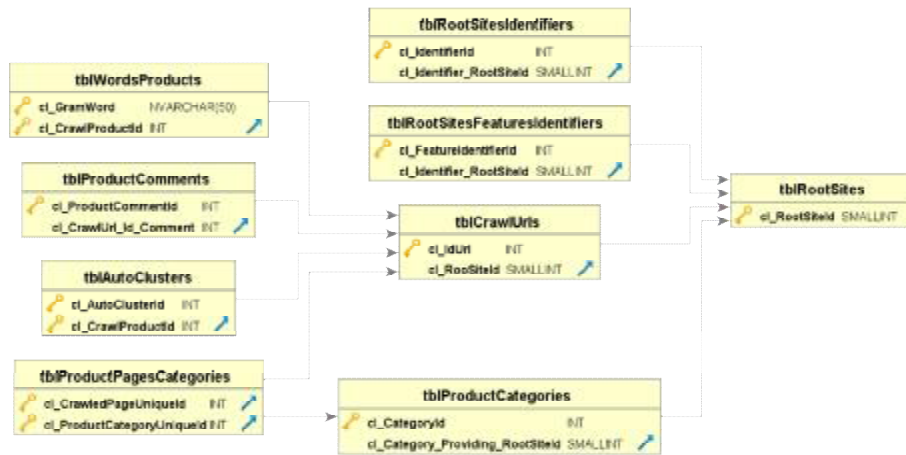


Figure 4.2. The tables used in EcCrawler

4.3. Initialization

Initialization process sets up the global variables; manages continuously running Tasks⁴ and functions; performs thread pool initialization; controls global unhandled exception handler; and runs close handlers.

URL Handler, Crawler, Pages Processor, and Global Statistics handler are Tasks that run continuously. These tasks continuously start new Tasks in a predetermined time interval, and they continue to operate until application termination.

4.3.1. Thread Pool Initialization

.NET framework determines the number of threads that is spawned automatically according to the application demand. However, this may yield poor results if the software is running on a powerful hardware when the hardware demand is high. To be able to start a pre-determined number of threads and utilize the system resources better, we propose to set the thread pool count manually before spawning the threads as shown in Appendix Code Snippet 1.

SetMaxThreads sets the maximum number of threads for the thread pool, and if there is more demand than the pre-determined maximum count, the demands are queued until the thread pool becomes available. *SetMinThreads* sets the minimum number of threads for the thread pool before switching an algorithm to manage the thread pool. In both functions that are shown in Appendix Code Snippet 1, the first parameter is the number of worker threads in the thread pool, and the second parameter is the number of asynchronous I/O threads in the thread pool. Worker threads are used for the active works done in the thread pool while I/O threads are more likely to wait for an external operation to complete such as a data receive from the network. We propose to set both *SetMaxThreads* and

⁴ Task Class. Represents an asynchronous operation. Microsoft Developer Network, <https://msdn.microsoft.com/en-us/library/system.threading.tasks.task>

SetMinThreads to an equal high number to obtain maximum available resources for the thread management system.

4.3.2. Exception Handling

Setting up proper exception handling enables developers to discover not properly handled errors during the program run. While our software is running in a real case scenario, we propose always to setup global exception handling to prevent from accidental application crashes. Moreover, errors are need to be properly logged for future analyses. Close handlers are also necessary to ensure that no data is lost when the application is terminated by running a command or clicking the close button at the right top of the UI. Moreover, close handlers can be set to run when an unexpected application termination happens for preventing any possible data loss. Code Snippet 2 given in the Appendix section shows how to setup global exception handling and application of close handlers in .NET framework.

4.3.3. UI Independent Tasks

Executing every task independently from the main thread can ensure application responsiveness, because none of the tasks would block the main thread. The best way to achieve this is to start a new independent *Task* for every process that application is going to do. We propose that only UI updates should run in the main thread by using polling methodology. Thus, *Task.Factory* method as shown in Appendix Code Snippet 3 can be used to spawn new threads. This method is one of the most optimal ways of doing multi-threading in C# WPF applications when using .NET 4.5 because *Tasks* are more lightweighted when they are compared to *Threads*⁵. Additionally, using *TaskScheduler.Default* parameter will ensure that all of the spawned *Tasks* will be independent of the main thread.

⁵ Task Parallelism (Task Parallel Library), [https://msdn.microsoft.com/en-us/library/dd537609\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/dd537609(v=vs.110).aspx)

4.3.4. Timer-Based Polling Methodology

We propose that polling methodology based system design provides both performance and robustness. Polling based system can be achieved by starting all of the tasks as different threads from the main thread during the initialization phase, and doing timer-based function calls in these child threads as shown in Appendix Code Snippet 4. *Timer*⁶ method also starts another *Task* and even if this spawned *Task* is terminated unexpectedly, the method continues to spawn new *Tasks*. Therefore, unexpected errors do not cause the application gets terminated, and unexpectedly terminated *Tasks* are properly eliminated.

4.4. Main Thread

Main thread is important for developers or clients to manage the crawling process. Displaying various statistics in the UI helps us greatly during the development of the crawler. Some developers may find that it is not essential for a web crawler to have a fancy UI. However, our proposed system can be applied to all kinds of software that require highly responsive UI. Running all of the tasks separate from the main thread can ensure UI responsiveness. However, updating the screen still requires accessing the UI thread. How to start polling threads from the main thread and refresh the interface properly without blocking the UI thread for a period of time are shown in Appendix Code Snippet 5. The *_timer* object ensures UI is updated with an interval and the *Dispatcher* event in this method ensures UI is not blocked. This methodology can minimize the risk of UI blockage.

In EcCrawler, a global static class is designed to store the events (e.g., URLs of the latest crawled pages, number of page-processing tasks that are running, how many crawling errors that happened, and so on). Polling threads constantly check this global static class and update the user interface without blocking the UI thread. However, even if this methodology is followed, UI freeze

⁶ Timer Class – MSDN, [https://msdn.microsoft.com/en-us/library/system.timers.timer\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.timers.timer(v=vs.110).aspx)

can still happen under certain situations. We noticed that this problem happens due to the GC (Garbage Collector) of the .NET framework. When the GC runs in the default mode which is *workstation*, if there are too many objects being constructed and destructed in the running software, it may pause all of the threads, including the main thread for a long time, and thus cause UI freeze, and software performance degrades. During the development of the EcCrawler, figuring out and solving this problem has taken some time. A possible solution to performance degradation issue is to set GC mode to *server* which does not pause all of the threads and support multi-threaded garbage collection. This setting is especially useful when the software is running on a system that has multi-core CPU. Code Snippet 6 given in the Appendix section shows how to set GC mode. After this mode change, according to our observations, even if thousands of *Tasks* run at the same time, UI responsiveness is not affected as long as there are sufficient system resources to update the UI. Also, changing the GC mode improves overall performance significantly. Experiments about this modification are presented in Section 8.1.5.

4.5. URL Handler

One of the main components of a web crawler is a URL handling system (Castillo, 2004). The main job of this system is to queue which URLs will be crawled next. Using MS-SQL server makes easy to build the URL handler system and it provides some benefits which are explained in the below paragraphs.

Several key aspects are observed when designing a URL handler system:

- i. Both the original URLs and their normalized versions that can be obtained by applying some methods like SHA256⁷, should be stored since some websites work with case-sensitive URLs.

⁷ <https://en.wikipedia.org/wiki/SHA-2>

- ii. Newly discovered URLs should be saved to the hard disk as batches in order to obtain better performance in terms of disk access time. Both versions of the URLs should be kept in a properly designed data structure such as *hashsets* or *dictionaries* until they are written to disk. Not-properly designed structures can reduce performance of the crawler.
- iii. Since different threads concurrently access to the data structure that store URLs, using *private readonly* objects to *lock* the data structure can ensure thread safety and data consistency. An example of such case can be seen in Appendix Code Snippet 7.
- iv. URLs that are failed to be crawled for a particular time should be disabled to prevent crawler trapping at the same URLs. Disabling the failed URLs must be done both in the database and in the URL repository of the software. In EcCrawler, we have set a variable which is called as retry count to 3 to solve this problem. In our crawler, after three consecutive failures, the URL is disabled for 24 hours. We did not conduct extensive experiments to find optimal parameters for these settings. Therefore, optimal value for retry count can be determined as future work.
- v. Another vital point is to provide load balance between the different E-commerce websites. Size and capacity of each site are distinct from each other, so the number of the crawling tasks for each website should be different and this value should be determined separately for each site to ensure both politeness and maximum performance of the crawling task. One of the major mistakes that were made during the early development stage of EcCrawler was the nonexistence of any load balancing system between the E-commerce websites. The URL handler was fetching the next batch of the crawling URLs from the database by the earliest discovery date. Simply doing breadth-first search (Najork & Wiener, 2001) which is one of the very first proposed approaches (Pinkerton, 1994) in the literature, caused that a few huge sized E-commerce sites get all of the

available crawling agents and thus decrease the overall application performance and violate the politeness rule of the crawling. One methodology proposed to solve this problem is to use the completion time of the previous crawling session (Najork & Wiener, 2001). By considering the crawling speed of each website individually, the number of agents for each website and the pause period for each consequent crawling can be individually determined. This approach requires determining threshold values for these parameters. This method favors very fast websites, and this may result in several enormous and very fast websites to get the majority of the available agents. One of the possible ways of solving this problem is to assign a unique static number for the maximum number of fetching agents at a time for each one of the E-commerce sites. When retrieving next crawling URLs from the database, a specified number of links for each E-commerce site is retrieved according to the pre-determined thresholds. The URL handler must also keep track of the list of URLs that are crawled currently to maximize the number of crawling agents for each website. When the crawler asks the URL handler to send the next batch of the queued links, a pre-determined number of URLs need to be provided for each website to accomplish load balancing. We defined different number of links to be crawled for the each website according to our empirical analysis of early crawling results. Additionally, public search engines can be used to determine these values. All the main search engines (e.g., Google, Yahoo, Bing, Yandex, and so on) support “*site:mysite.com*” queries which provide the number of estimated results each of which can be considered as a different page. Although these results are not very accurate and reliable (e.g., they don’t display exact statistics, or some sites could have duplicate pages because of incorrect URL structures, and so on), they can be used to roughly estimate the size of the crawled websites and determine the number of crawling agents proportionally for each site

(e.g., if there are 100 available agents at a time, distribute them according to these size values).

- vi. After load balancing strategy is decided, the question arises which links should be crawled next. Several different techniques have been proposed in the literature to determine which links should be the next. These methods can be summarized as follows: (a) First discovered is first crawled (Najork & Wiener, 2001); (b) Giving priority by using the count of incoming hyperlinks (Cho, et al., 1998) to crawl more relevant pages first: This technique is now obsolete due to the massive amount of link spamming; (c) Giving priority by some page scoring such as PageRank (Cho, et al., 1998; Cho & Garcia-Molina, 2003; Cho & Schonfeld, 2007; Abiteboul, et al., 2003; Baeza-Yates, et al., 2005; Chien, et al., 2004). However, most of the advanced page ranking algorithms may perform poorly against current web pages because of the newly developed advanced search ranking manipulation techniques⁸ (e.g., link schemes, sneaky redirects, affiliate programs, paid links, and so on). Google uses over 200 different factors⁹ to determine the PageRank of web pages today to fight against these techniques and deliver high-quality search results. Since in our work, the websites that are going to be crawled are pre-determined, and there is a finite number of pages; we did not use any page ranking algorithm to give priority to pages which are crawled first. Instead of some page ranking mechanism, we have employed manually constructed filtering rules to prevent crawling unnecessary pages. For example for the E-commerce site VatanBilgisayar, if the newly discovered URLs contain “/webapp/” or “searchQuery=” strings, they are discarded. Because, a URL, which

⁸ Quality guidelines – Google,
<https://support.google.com/webmasters/topic/6001971>

⁹ Google Inside Search, Algorithms,
<https://www.google.com/insidesearch/howsearchworks/algorithms.html>

contain these two strings, cannot be a category page or a product page. For our task, we only need to crawl and process the category pages which list the product pages, and the product pages themselves. Currently, a domain expert analyses the discovered URLs and decides which strings are used for filtering the web pages. Whenever structures of the product and category pages are changed, we need to update our rules which can determine the product and category pages. Therefore, automatic determination of product pages and category pages is needed and this can be done by employing a classifier. As our main aim in this study is to show .NET framework implementation based solutions for the problems encountered, we plan to include a classifier to our crawler as a future work.

4.6. URL Crawler

URL crawler (i.e., page fetching) is another core part of a web crawler (Olston & Najork, 2010). Tasks of a URL crawler can be summarized as follows: crawling the given link, extracting the hyperlinks from the crawled page, and returning both the extracted links and the source code of the crawled page. In the EcCrawler, only product pages are processed so the crawler checks whether the page is a product page or not. If it is not a product page, the source code is not saved. Also, all of the extracted URLs are processed by applying techniques like “replace words”, “ignore words”, “trimming”, “not allowed types” (e.g., PNG, JPG, PDF, and so on), “disallowed by robots protocol”, and so on. Some of these techniques are manually defined after inspection of the targeted E-commerce websites. Eliminating unnecessary pages (e.g., duplicate pages, irrelevant pages, etc.) from processing makes a huge impact on the overall performance of crawling. When the crawler tasks complete their operation, they return the discovered URLs and the source of the crawled page to the URL handler so that they can be saved in the database system; and the discovered new URLs can be added to the crawling queue.

The performance of the web crawlers is highly depended on the used settings. However, in most cases these settings are not mentioned in the related literature (e.g., (Shkapenyuk & Suel, 2002; Olston & Najork, 2010; Edwards, et al., 2001)). To possibly obtain better performance, we advise using *HttpWebRequest* class in .NET framework because this class allows tuning of many different parameters. From these parameters *Headers*¹⁰, *Encoding*¹¹ of the *StreamReader*, *KeepAlive* and the *ResponseUri* are necessary. Headers are used to enable HTTP compression¹² when fetching pages from the server and thus significantly reduce the network usage. HTTP compression is achieved by adding *gzip*¹³ and *deflate*¹⁴ parameters to the headers by using “*Accept-Encoding*” key as shown in Appendix Code Snippet 8. *Encoding* of the stream is essential for parsing of the fetched source code properly. It is provided to the *StreamReader*¹⁵ function, and it determines the character encoding of the fetched source code¹⁶. *Keepalive*¹⁷ feature may significantly affect the overall network performance according to the crawling scenario. When the *Keepalive* feature is enabled, it uses a persistent HTTP connection¹⁸ to the crawled server for multiple HTTP requests instead of establishing a new connection session when each time a page is fetched. The only

¹⁰ Headers, [https://msdn.microsoft.com/en-us/library/system.net.httpwebrequest.headers\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.net.httpwebrequest.headers(v=vs.110).aspx)

¹¹ Encoding Class – MSDN, [https://msdn.microsoft.com/en-us/library/system.text.encoding\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.text.encoding(v=vs.110).aspx)

¹² HTTP compression – Wikipedia, https://en.wikipedia.org/wiki/HTTP_compression

¹³ Gzip – Wikipedia, <https://en.wikipedia.org/wiki/Gzip>

¹⁴ DEFLATE – Wikipedia, <https://en.wikipedia.org/wiki/DEFLATE>

¹⁵ StreamReader – MSDN, [https://msdn.microsoft.com/en-us/library/system.io.streamreader\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.io.streamreader(v=vs.110).aspx)

¹⁶ Character encoding – Wikipedia, https://en.wikipedia.org/wiki/Character_encoding

¹⁷ KeepAlive – MSDN, [https://msdn.microsoft.com/library/system.net.httpwebrequest.keepalive\(v=vs.110\).aspx](https://msdn.microsoft.com/library/system.net.httpwebrequest.keepalive(v=vs.110).aspx)

¹⁸ HTTP persistent connection – Wikipedia, https://en.wikipedia.org/wiki/HTTP_persistent_connection

disadvantage of *Keepalive* is, it causes extra memory usage (especially for the web server) therefore, when there is multiple access to the server, the *Keepalive* feature should be enabled.

Also, there will be many pages that redirect to another page. Therefore, it is important for a crawler to handle the redirected URLs for achieving proper crawling and preventing duplicate crawling. *HttpWebRequest* class by default allows automatic redirections. So if any redirection happens during the HTTP request to the server, it is handled by *HttpWebRequest* class automatically, and the final crawled URL can be obtained from *ResponseUri.AbsoluteUri* parameter of the *WebResponse* class.

All settings can be done as shown in Appendix Code Snippet 8. Also, EcCrawler shows respect to the canonical URLs¹⁹ system of Google. Canonical URLs show absolute links of pages to prevent duplicate pages²⁰. If the source code has canonical URL parameter and the URL of the fetched page does not match with the canonical URL, then the URL handler only returns the canonical URL of the fetched page. If this returned canonical URL has been seen before by the crawler, the fetched URL is marked as “*duplicate by canonical*”. For example, assume that the discovered URL is “http://www.buroteknik.com/Cross-883-3-Atx-Bazalt-Siyah-Versatil_151274.html” while the canonical URL in the page source is “http://www.buroteknik.com/en-ucuz-cross-883-3-atx-bazalt-sc4b0yah-versatc4b0l-fiyati-ozellikleri_151274.html”. As can be seen, the URLs are different however they are the same pages.

One other critical problem is infinite-URLs generation. Some websites use bad link structure which results in generating infinite amount of URLs. Infinite-URLs generation happens by adding more directory level to the URLs. The following URL can be given as an example of such case:

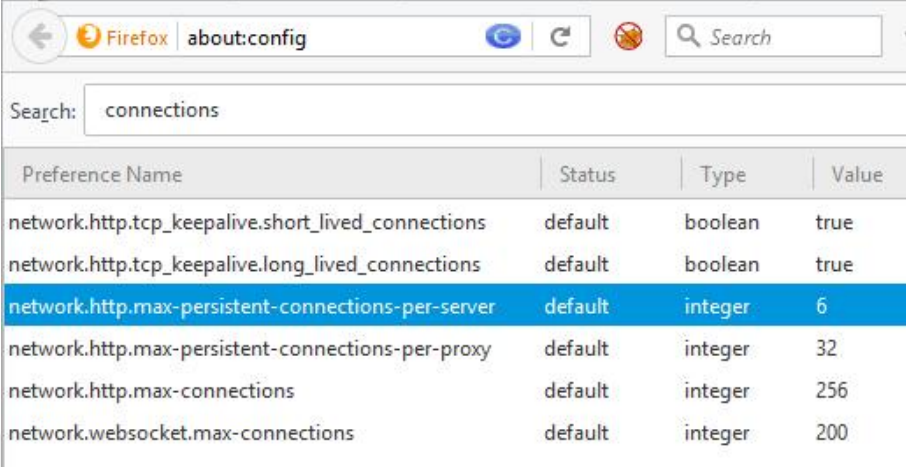
¹⁹ Use canonical URLs, <https://support.google.com/webmasters/answer/139066>

²⁰ Duplicate content – Google, <https://support.google.com/webmasters/answer/66359>

<http://www.mysite.com/toys/computers/toys/computers>. To possibly overcome this problem, a maximum directory depth for each website can be set manually after a domain expert inspects URL structure of each website. However, this may cause false rejects if the URL itself contains “/” character instead of for directory level. On the other hands, the gains that have been obtained by setting a max directory depth value are much greater than these very rare false rejects when developing EcCrawler. As many E-commerce websites generate an infinite number of links they cause crawling of too many duplicate pages.

In today’s web, there is a vast amount of web pages to crawl and the number of web pages also increases exponentially. To crawl all these pages, more concurrent connections are needed to be opened to the same host by spawning multiple crawling agents. However, when the default settings are used, .NET framework only allows having two connections concurrently to the same server. Two connections concurrently are the default limit defined by HTTP/1.1²¹ protocol in 1997. At that time, when the capabilities of web server hardware and software are taken into account, this limit can be considered as reasonable. However, nowadays, the power of web server hardware and software has significantly increased. Additionally, sizes of websites have been exponentially increased. As a consequence of these stated reasons, to obtain better crawling performance we need to use multiple crawling agents to the same host, therefore this limit needs to be increased. Moreover, many of the modern web browsers already allow more than two persistent connections for each host (e.g., Mozilla Firefox allows six current connections by default as it can be seen in Figure 4.3.). Concurrent connection limit per host can be increased by setting up *DefaultConnectionLimit* parameter as shown in Appendix Code Snippet 9 for .NET framework when using C#. This parameter must be set before the web request is made hence; EcCrawler sets this parameter at the start of the application.

²¹ Hypertext Transfer Protocol -- HTTP/1.1,
<https://www.w3.org/Protocols/rfc2616/rfc2616.html>



Preference Name	Status	Type	Value
network.http.tcp_keepalive.short_lived_connections	default	boolean	true
network.http.tcp_keepalive.long_lived_connections	default	boolean	true
network.http.max-persistent-connections-per-server	default	integer	6
network.http.max-persistent-connections-per-proxy	default	integer	32
network.http.max-connections	default	integer	256
network.websocket.max-connections	default	integer	200

Figure 4.3. Default max persistent connections per server settings for Mozilla Firefox version 44.0.2

For further analysis, we have examined statistics of the top 10 Turkish E-commerce websites in Turkey by using Alexa²² and Google services to see whether maximum two simultaneous persistent connection for each host is suitable or not. Our aim in this analysis is to test whether these targeted E-commerce websites' daily fresh content could be obtained while obeying concurrent connection count limitation of the HTTP/1.1 protocol. So estimated index size for each website is collected by querying "*site:mystore.com*" from Google. Then, average loading speed of each of these websites are gathered from Alexa which calculates page loading speed by computing how long does it take the user's browser to load DOM (i.e., the structure of the page which does not include images or CSS styles) of the page. Loading speed of browser can be slower than page source fetching of the crawler however, since Alexa takes only DOM loading into consideration and has enormous statistical data about these websites, we believe that this measure can give an overall idea. According to Cho & Garcia-Molina (2000) more than 40% of

²² An Amazon company that provides public statistics about websites, <http://www.alexa.com/>

the pages in the .com²³ domain changes every day. So to keep the freshness of the E-commerce websites, we have calculated minimum number of daily page crawling requirement for each site and this is computed by multiplying the estimated page count of the site by 40%. Moreover, we have calculated how many pages can be crawled for each E-commerce website as follows:

$$HPCBC(\epsilon) = \frac{H \times M \times S \times CC}{ALS(\epsilon)}$$

where $HPCBC(\epsilon)$ is the number of pages that can be crawled for the selected E-commerce website in a day, H is hours in a day (i.e., 24), M is minutes in a hour (i.e., 60), S is seconds in a minute (i.e., 60), CC is the concurrent connection count (i.e., 2), and $ALS(\epsilon)$ is the average load speed in seconds of the selected E-commerce website which is directly taken from the Alexa's statistics. The results are displayed in the Table 4.1 where estimated index size is provided by Google and the coverage is $\frac{PCD(\epsilon)}{HPCBC(\epsilon)}$ where $PCD(\epsilon)$ is the minimum # of page to be crawled per day to re-crawl the updated content each day.

²³ COM = Commercial, any commercial related domains meeting the second level requirements, <http://tools.ietf.org/html/rfc920#page-2>

4. PROPOSED FOCUSED WEB CRAWLER SYSTEM Furkan GÖZÜKARA

Table 4.1. Statistical analysis of concurrent connection limit on E-commerce websites

E-commerce Site	Estimated Index Size	Average Load Speed ($ALS(e)$)	# of Pages to be Crawled per Day ($PCD(e)$)	How Many Pages can be Crawled per Day ($HPCBC(e)$)	Coverage
Gittigidiyor	788,000	1.682	315,200	102,734	%32.59
N11	781,000	1.079	312,400	160,148	%51.26
Hepsiburada	778,000	1.605	311,200	107,663	%34.59
Vatanbilgisayar	3,230,000	1.758	1,292,000	98,293	%7.60
Trendyol	570,000	2.053	228,000	84,169	%36.91
Teknosa	1,070,000	3.153	428,000	54,804	%12.80
Markafoni	473,000	1.999	189,200	86,443	%45.68
Sanalpazar	552,000	0.986	220,800	175,253	%79.73
Kitapyurdu	627,000	1.088	250,800	158,823	%63.32
Tozlu	1,110,000	1.141	440,000	151,446	%34.41

When the results in Table 4.1 are analyzed, it can be concluded that two concurrent connections at a time are not enough to keep up daily content freshness in today's web. However, these are rough estimations, and detailed experiments may be required to obtain more accurate results because of several reasons:

- Google statistics are not entirely accurate thus, there may be more pages;
- Statistics that are provided by Alexa are estimations of various parameters, and they may not be very accurate;
- Page crawling speed is highly depended on the network connection speed and the time of the day when the web server is accessed;
- It is not possible to crawl only updated pages without re-crawling the other pages as well;
- 40% update estimation is based on the experiments conducted in 1999 and today it may be very different percent;
- In some cases not daily but hourly, even minutely freshness may be required, which means crawling for more pages may be necessary;
- There are some websites that contains much more content and thus requires much more frequent crawling sessions (e.g., Amazon²⁴ has 161,000,000 estimated index size, and so on).

Even though these statistics are not perfect, it is clear that only two concurrent connections is not sufficient currently, and it should be increased according to the tasks that are running, and the system resources used by the tasks. Therefore, we have set this limit to a large number which is equal to 1000. On the other hands, to prevent any bottleneck on the E-commerce websites, we make sure that we have never exceed 50 concurrent connections, as our maximum crawling agent count per website has been set to 50. We have set this number based on our empirical analysis of the crawling speed by testing several different values.

4.7. Database Communication

In EcCrawler, all of the database queries are executed by using a single database handler class. Using single class makes database handling system easier to

²⁴ <http://www.amazon.com/>

4. PROPOSED FOCUSED WEB CRAWLER SYSTEM Furkan GÖZÜKARA

manage. Several different functions are written to handle specific cases according to EcCrawler requirements. One of the database management functions designed for EcCrawler is shown in Appendix Code Snippet 10. In the displayed function there are several key parameters:

- when *integrated security=SSPI* is used, no user ID or password is required since the Windows's current credentials is used for authentication;
- *Max Pool Size=20000* sets the connection pool size of the SQL-Server for that client and if the application opens a lot of concurrent connections to the SQL-Server this parameter should be set to a high number;
- enabling *Pooling*²⁵ improves overall performance by preventing from opening a physical channel each time a new connection is opened;
- *Connection Timeout* determines the timeout of connection opened before the connection is automatically closed and this should be set to an appropriate value according to the scenario (e.g., if exhausting queries are being executed, this timeout should be set to a high number in order to prevent connection problems);
- *CommandTimeout* is a different parameter than the *Connection Timeout* thus it should also be set to an appropriate value.

Performance improvement of this function comes from using a single database connection for executing multiple updates or insert operations. The function gets a list of parameters and objects of multiple queries and then save them in the database by using *SqlCommand*. This kind of batch inserts and updates may improve overall performance by preventing multiple server connection authentications. To minimize the number of server connection authentication, a

²⁵ SQL Server Connection Pooling, [https://msdn.microsoft.com/en-us/library/8xx3tyca\(v=vs.100\).aspx](https://msdn.microsoft.com/en-us/library/8xx3tyca(v=vs.100).aspx)

single static connection is shared among all of the threads to prevent authentication overhead. However, SQL connection is not thread safe and need to be locked each time it is accessed. Lock methodology can decrease overall performance when there are too many threads and, therefore, it is better to use a new connection for each thread.

We present experiments on the effects of batch access to database and static connection performance in Section 8.1.1. Additionally, it is a good practice to update the same tables from the same threads always. This methodology can prevent crucial table locks and improve the overall performance. However, *select* statements, in general, are directed to the same table from different threads. Thus, there can be occasions that both *select* and *update* statements are executed on the same table at the same time from possibly different connections. In such case, “*transaction was deadlocked on lock resources with another process and has been chosen as the deadlock victim*” error may happen and SQL server always undoes the least amount of work required by the transaction. By default, priority of *select* transactions is lower than priority of *update* transactions, so *select* transactions are denied. The possible solution to this problem is to check SQL server errors and if there is a resource deadlock victim error when the *select* statement is executed, the same *select* statement is repeated a pre-determined number of times with a delay. Moreover, every unexpected error should be logged and handled in a proper way.

4.8. Pages Processor

Pages processor class processes pages, and translates the semi-structured data into the structured form. These specific processors are also known as wrappers (Kushmerick, 1997) in Information Retrieval (IR) area and they are one of the most important parts of the focused web crawlers. General web crawlers also have page processing tasks (e.g., for link extraction) however, when focused web crawling is made, the task is usually much more complex because the aim of focused crawling is to obtain structured data from unstructured targeted domains.

4. PROPOSED FOCUSED WEB CRAWLER SYSTEM Furkan GÖZÜKARA

Commercial systems usually require very high success rate of obtaining targeted data. Obtaining such success rates by using automatic or semi-automatic wrapper generation methodologies (e.g., (Doorenbos, et al., 1997; Yang, et al., 2000a)) are very hard to achieve in practice today because of various reasons: (i) browsers have become so robust in the recent years that they display even very erroneous HTML source data correctly and thus data are not presented as structured as before; (ii) complexity of the web pages has significantly increased and the amount of ordinary structured data (e.g., table usage, etc.) has decreased. Therefore, in order to achieve a very high success rate, we decided to develop the wrappers manually for EcCrawler and we obtained very high success rate (i.e., about 100%). As our crawler only visits a pre-determined set of e-commerce websites, we can develop some rules by manually inspecting the source codes of the product pages to extract information about the products. When the manual wrapper crafting methodology is chosen, the system design is extremely important for rule generation. Proper system design makes the rule generator's job extremely easy and improves the software performance. For example, for VatanBilgisayar E-commerce website, if an object, that satisfies the below conditions, exists in the source code of the crawled page, the page is determined as a candidate product page and it is processed by page processor agent:

- HTML Object Type: span
- HTML Object Identifier: class
- HTML Object Identifier Name: aktKod

Main task of pages processor is to translate the desired information from raw HTML source code into a structured form by using the wrappers. For this task,

regular expressions²⁶ (regex) can be utilized. However, due to the erroneous nature of the source codes and the complexity of the regexes that are required, using an advanced HTML parsing library such as HtmlAgilityPack²⁷ can be a better solution. HtmlAgilityPack is capable of fixing HTML errors and generate properly structured DOM (Document Object Model)²⁸. It is open source and written in .NET framework. EcCrawler's wrappers use HtmlAgilityPack for information extraction task. The system design is completely up to the developer. For EcCrawler, XPath²⁹ queries are defined for each E-commerce website and executed iteratively to get all of the necessary information (e.g., product title, product categories, etc.) in a structured way. There are several key aspects when building HTML parser with HtmlAgilityPack framework. *HtmlDocument* and *HtmlNode* are one of the most commonly used classes and they are not *System.IDisposable*. Because of this reason, setting them to *NULL* when the object is not necessary anymore, improves the overall performance. One other important aspect is, to check whether an object is *NULL* or not before using it due to the unpredictable nature of the web crawlers. Additionally, in some cases, parsing JSON (JavaScript Object Notation)³⁰ data is also necessary. The usage of JSON significantly increases performance by providing more asynchronous pages and better user experience. However, HtmlAgilityPack is not capable of parsing JSON data. For parsing JSON data, another open source and free library Json.NET³¹ is used.

²⁶ Regular expression – Wikipedia,
http://en.wikipedia.org/wiki/Regular_expression

²⁷ “*HtmlAgilityPack is an agile HTML parser that builds a read/write DOM and supports plain XPATH or XSLT*”, <https://htmlagilitypack.codeplex.com/>

²⁸ Document Object Model – Wikipedia,
http://en.wikipedia.org/wiki/document_object_model

²⁹ XML Path Language (XPath) – W3C, <https://www.w3.org/TR/xpath/>

³⁰ <http://www.json.org/>

³¹ Popular high-performance JSON framework for .NET,
<http://www.newtonsoft.com/json>

4.9. Collaboration and Resource Sharing Between Different Threads

Collaboration and resource sharing between different threads can be done either by using a database system as a middleware or having public functions to access private variables. Using the database as middleware can degrade the system performance due to connection authentication overhead and accessing to the hard disk instead of using RAM. Resource sharing between different threads requires data access to the same objects from multiple threads at the same time. Collaboration between multiple threads can be ensured by properly locking the non-thread-safe objects before accessing them.

There is a dedicated class in EcCrawler, which holds statistics and events' logs to show on the user interface and save to the hard disk. For keeping events' logs, *private static List* objects are used and accessed via public functions. For statistics such as how many pages are crawled, *public static long* objects are used. However, operations on the numerical objects (e.g., integer, long, etc.) are also not-thread-safe, and thus, they need to be either locked or more properly *Interlocked* method is needed to be used when accessing them for both read or update. Also, to ensure thread safety, it is a necessity to use the same locking object for the same non-thread-safe objects. If multiple objects are used for locking to ensure thread safety of a non-thread-safe object, this may result in either error or data inconsistency. It may even cause deadlocks when a thread locks the first object and at the same time, another thread locks the second object, and no locks are released until the same thread locks both of the objects. Code Snippet 11 given in the Appendix section shows an example usage of *Interlocked* and object-based locking method.

4.10. Error Log System

In software development, unexpected errors can always happen and in the worst cases they may cause unexpected application terminations. Also it is very hard to test the application for all cases and fix all of the possible errors before the

public release. In order to minimize these possible errors, software developers tend to follow software release life cycle³² (e.g., alpha, beta, release candidate, etc.). Therefore, logging every error, and handling unexpected errors are crucial and important. In EcCrawler, an entire class is designed for error-logging. This class uses *StreamWriter*³³ class to log errors into the text files. The database system can also be used to log errors however for faster logging we have used text files. *StreamWriter* object is not a thread safe object and does not *Flush*³⁴ after every write automatically by default. So locking is necessary when accessed by multiple threads. Also, it has to be either manually flushed or auto-flush must be set to true not to lose any data in case of unexpected application terminations.

4.11. Thread Count Management System

The thread count management system is an important part of developing a multi-threaded software. The importance increases when particular tasks need a certain number of active threads constantly. For example, to keep a steady number of active crawling agents for each host individually, we propose that one possible solution is to use the polling methodology by timer based checks. In this method, spawned threads are kept in a thread list. Whenever the timer ticks, the number of unfinished threads is checked, and the required number of new threads are started as shown in Appendix Code Snippet 12. This strategy properly ensures that at any given time there exist a certain number of active threads for the task. For multi-threading in .NET 4+, two different classes which are Task³⁵ and Thread³⁶ classes

³² Software release life cycle – Wikipedia, https://en.wikipedia.org/wiki/Software_release_life_cycle

³³ StreamWriter Class – MSDN, [https://msdn.microsoft.com/library/system.io.streamwriter\(v=vs.110\).aspx](https://msdn.microsoft.com/library/system.io.streamwriter(v=vs.110).aspx)

³⁴ “Clears all buffers for the current writer and causes any buffered data to be written to the underlying stream”³³

³⁵ Task Class – MSDN, [https://msdn.microsoft.com/en-us/library/system.threading.tasks.task\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.threading.tasks.task(v=vs.110).aspx)

4. PROPOSED FOCUSED WEB CRAWLER SYSTEM Furkan GÖZÜKARA

can be used. Managing and starting Tasks are relatively easier and have better performance³⁷ than using Threads. Thus, Task class is preferred starting from .NET 4+ when writing multi-threaded applications. Therefore, EcCrawler uses Task class instead of Thread class.



³⁶ Thread Class – MSDN, [https://msdn.microsoft.com/en-us/library/system.threading.thread\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.threading.thread(v=vs.110).aspx)

³⁷ Task Parallelism – MSDN, [https://msdn.microsoft.com/en-us/library/dd537609\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/dd537609(v=vs.110).aspx)



5. PROPOSED CLASSIFICATION SYSTEM for SENTIMENT ANALYSIS**5.1. Dataset of Sentiment Analysis**

In this study, we have developed our own dataset by crawling the top 50 popular E-commerce websites in Turkey. Then, comments on products are fetched by our focused crawler. Table 5.1 shows some statistics about the comments that are crawled by our system. From 2,214 comments that have 49 or below user score, 1,000 comments are randomly picked as negative comments. From 42,234 comments that have 50 or above user score, 1,000 comments are randomly picked as positive comments. Next, a domain expert has manually analyzed all of the comments and decided which ones truly express an opinion. However, alone user scores were not enough to correctly classify comments as positive or negative. Thus, the domain expert also correctly classified the wrongly scored user comments. Then totally unrelated and neutral comments are removed from the dataset. The final comment dataset is composed of 600 positive and 600 negative comments. We have to note that, we did not make any categorical selection. So these comments are from a wide and very different category of products such as electronic devices, adult products, or perfumes. There are also many comments in the dataset that contains both positive and negative opinions since only fully neutral and unrelated comments are removed.

Table 5.1. Statistics of all comments that are crawled by our focused crawler system

# of comments without any user score	# of comments having user score 49/100 or below	# of comments having user score 50/100 or above	total # of comments crawled
37,215	2,214	42,234	81,663

5. PROPOSED CLASSIFICATION SYSTEM for SENTIMENT ANALYSIS

Furkan GÖZÜKARA

In our experimental evaluations we have also used popular publicly available datasets that are English movie review dataset (Pang & Lee, 2004) which has 1000 positive and 1000 negative comments; a greater sized English movie review dataset (Pang & Lee, 2005) which includes 5330 positive and 5331 negative comments; Turkish movie review dataset (Demirtas & Pechenizkiy, 2013) that has 5326 positive and 5326 negative reviews; English product review dataset that has reviews from books, DVD, electronics, and kitchen categories such that each category has 1000 positive and 1000 negative comments; and finally Turkish product review dataset (Demirtas & Pechenizkiy, 2013) having comments for books, DVD, electronics, and kitchen categories and each category has 700 positive and 700 negative comments. Preprocessed versions of all datasets that are used in this study are published on the GitHub³⁸ to be used by other researchers in their future studies to make more accurate comparison with our results.

5.2. The Classification System

To conduct more rational experiments, we split the dataset into 10 disjoint sets and apply 10-folds cross-validation for evaluation. In 10-fold cross validation, data is split into non-overlapping, equal sized 10-sets and tests are conducted 10-times with order while leaving one set out as a test set and using the rest of the sets as training set (Kohavi, 1995). Then the macro-average of the obtained scores from the tests is calculated, and it becomes the final score. To evaluate classification performance we use both accuracy and F-measure scores. Accuracy is the percentage of correctly classified instances in the test set. F-measure is the harmonic mean of precision and recall of classification of test instances, where precision is the exactness, and recall is the completeness of the classification algorithm.

³⁸ <https://github.com/FurkanGozukara/Sentiment-Analysis>

5. PROPOSED CLASSIFICATION SYSTEM for SENTIMENT ANALYSIS

Furkan GÖZÜKARA

Our system first applies pre-processing to the dataset. In the pre-processing phase, comments are converted into BOW form. We did not apply any stemming or stop word removal since it is shown to be not effective in sentiment analysis for the Turkish language (Parlar, 2016). We have converted all of the characters into lower case, discarded single letter words, replaced punctuations with a space character, applied transformation of diacritics and then removed non-ASCII characters. To the best of our knowledge, transformation of diacritics and removal of the non-ASCII characters at the same time is not done before in the Turkish sentiment analysis area. It is critical to note that removal of non-ASCII characters process should not be applied alone or before the transformation of diacritics process. If it is applied, it can cause unintended results such as “*alışveriş*” becomes “*alveri*”. Additionally, we have conducted experiments with N-gram transformation. N-character slicing can be replaced with N-words slicing as well however we have not applied word level N-gram transformation.

After preprocessing is done, features are assigned numerical weights and document vectors are formed since SVMs works with numerical values. This process is very crucial part of text classification task according to results of our empirical analysis when SVMs are used. How well the features are weighted directly affects the success of the system. We have applied a combination of different weighting metrics, and they are presented in Section 3.4.

After data is preprocessed and document vectors are formed, it is properly formatted for LibSVM. LibSVM accepts sparse data, and that is a big advantage in text classification where there can be thousands of features missing in each document. After data preparation is done, the C parameter is estimated by using the training set. The C parameter defines how much we want to avoid misclassifying the training dataset. There is a practical guide to usage of SVMs published by Hsu and colleagues (2010) and we refer to their study for information about SVM parameters. For determining the best C parameter, we use “grid-search” technique. For not overfitting the training model, we employ 10-fold cross-validation while

5. PROPOSED CLASSIFICATION SYSTEM for SENTIMENT ANALYSIS

Furkan GÖZÜKARA

using the training set. To find out best performing C parameter, we test the following C parameters and choose the best performing one: 0.03125, 0.0625, 0.125, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384. Our system is developed in C# programming language by using .NET 4.5 framework and LibSVMSharp³⁹ wrapper.



³⁹ <https://github.com/ccerhan/LibSVMsharp>

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

In standard HAC algorithm, the closest pairs are merged until the desired cluster structure is obtained (Rokach & Maimon, 2005). However, the most controversial part of the HAC is when we obtain the desired structure so that we can stop merging. Numerous techniques have been proposed for this problem. However, to our knowledge there does not exist a widely accepted solution for this problem. One of the most commonly employed techniques is to draw a Cluster Dendrogram as shown in Figure 6.1 and manually decide where to cut the clustering. Providing the number of desired clusters is another technique usually employed. A more detailed information about clustering techniques can be found in the survey published by Berkhin (2006). However, the mentioned two approaches and many of the other existing solutions require manual analysis or the number of desired clusters. In our problem, both cases are usually impossible to employ. Thus, we propose a new framework for doing Dynamic/Incremental HAC. In our framework, we do not employ stopping criteria, rather, we use distance thresholding along with multiple matching/blocking approaches until all records are processed.

Our system works recursively until no-new data is received and no new cluster emerges. It means that, when we say “the data”, it means either the new data and the candidate records that it can emerge a new cluster or it is a newly emerged cluster and the previous candidate clusters that it can merge again.

Our proposed system is composed of four stages. In the first stage, the data is preprocessed. In the preprocessing stage the following tasks are employed in the given order which we will explain in the following paragraphs:

- i. Diacritics are transformed into non-diacritic form.
- ii. Special word addition system is employed.

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

Furkan GÖZÜKARA

- iii. Special string replacements are applied.
- iv. Punctuation marks are removed.
- v. Special semantic normalizations are applied.
- vi. Non-ASCII characters are removed.
- vii. Lower case conversion is applied.
- viii. Character based N-gram transformation is applied.
- ix. Words that have lengths which are below 2 or above 30 characters are discarded.
- x. Finally, the title of the product is transformed into vectoral form.

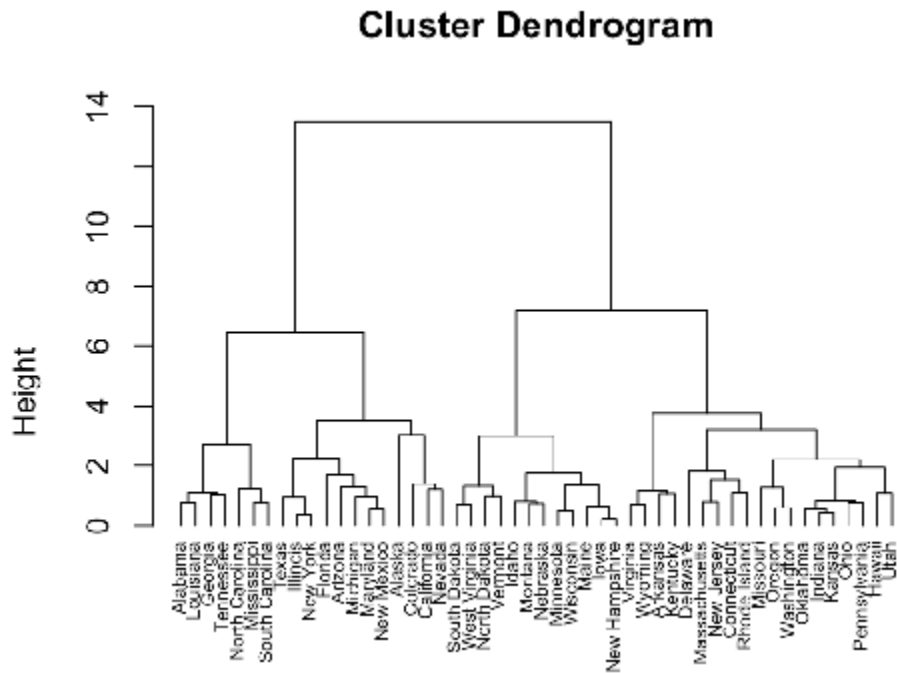


Figure 6.1. An example of a cluster dendrogram (Kassambara, 2015)

An example of special string replacement is replacing " with "inc". The special string replacement list is manually generated by the domain expert and presented in Table 6.1 where £ represents the space character.

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

Furkan GÖZÜKARA

Table 6.1. Special string replacement list

Original String													Replacement String
\				"			inch			`			inc
\n	\r	,	//	/	!	'	;	-	£.	.£	(	)	* _
													£

An example of semantic normalization is replacing “*toner*” with “*kartus*”. The semantic normalization list is manually generated by the domain expert and presented in Table 6.2.

Table 6.2. Semantic word replacement entire list

Original Word	Semantic Replacement
white	beyaz
black	siyah
red	kirmizi
blue	mavi
gold	altin
toner	kartus

The special word addition system is our novel proposal. After extensive experimenting and research, we have discovered that there are three special characters as “-”, “.”, “:” for product titles. We split tokens from these characters and add split parts along with the special character removed version of the token. For example token “11-n000nt” becomes “11”, “n000nt” and “11n000nt”. The benefit comes mostly at the product codes which are usually written in different formats in different E-commerce websites. When employing character based N-gram transformation, we have kept the original title as well. So our method uses original title plus character based N-gram transformed version.

A full example for data preprocessing step by step is presented in Table 6.3.

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

Furkan GÖZÜKARA

Table 6.3. An example of data preprocessing steps in record linkage task

Step	Product Title
Raw	HP Pavilion 11-n000nt Pentium N3540 4GB 750GB 11.6" Dokunmatik Win8 - K0W08EA Dizüstü Bilgisayar
Diacritics are removed	hp pavilion 11-n000nt pentium n3540 4gb 750gb 11.6" dokunmatik win8 - k0w08ea dizustu bilgisayar
Special word addition is applied	hp pavilion 11n000nt pentium n3540 4gb 750gb 116" dokunmatik win8 k0w08ea dizustu bilgisayar 11 n000nt 11 6"
Special string replacements and punctuation removal are applied	hp pavilion 11n000nt pentium n3540 4gb 750gb 116inc dokunmatik win8 k0w08ea dizustu bilgisayar 11 n000nt 11 6inc
Non-ASCII characters are removed	hp pavilion 11n000nt pentium n3540 4gb 750gb 116inc dokunmatik win8 k0w08ea dizustu bilgisayar 11 n000nt 11 6inc
Min or max length filter is applied	hp pavilion 11n000nt pentium n3540 4gb 750gb 116inc dokunmatik win8 k0w08ea dizustu bilgisayar 11 n000nt 11 6inc
Original plus 2-gram transformation is applied	hp pavilion 11n000nt pentium n3540 4gb 750gb 116inc dokunmatik win8 k0w08ea dizustu bilgisayar 11 n000nt 11 6inc hp pp pa av vi il li io on n1 11 1n n0 00 00 0n nt tp pe en nt ti iu um mn n3 35 54 40 04 4g gb b7 75 50 0g gb b1 11 16 6i in nc cd do ok ku un nm ma at ti ik kw wi in n8 8k k0 0w w0 08 8e ea ad di iz zu us st tu ub bi il lg gi is sa ay ya ar r1 11 1n n0 00 00 0n nt t1 11 16 6i in nc

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

Furkan GÖZÜKARA

In the second stage, after the product records are preprocessed, they are loaded into the dictionaries. At this stage, the frequency of each word in the product title is calculated and saved as raw-frequency. Now we have vector representation (Salton, et al., 1975; Singhal, 2001) of the each product title where the weights are raw-frequency. Since many of the string based algorithms works only on the numerical vector representation of the strings, we have chosen this methodology. Additionally, it is possible to improve the success rate significantly by employing the right weighting scheme. However, we still keep original preprocessed form of the words as well for certain functions. Once the data is ready, the third stage starts.

In the third stage, the system starts to cluster records one by one since our system does online (incremental) clustering. When applying matching/blocking functions, we always think each cluster as a single record thus, all features in each cluster are considered. First candidate cluster set is generated for the new record by applying Traditional Blocking where a certain number of intersecting feature count is necessary to be counted. When N-gram transformation is applied, the effectiveness of Traditional Blocking significantly reduces since each record has more features, and this should be considered. Once the candidate cluster set is obtained, the vector weighting function is called since we will calculate the distance between cluster pairs. When distances between clusters pairs are calculated, different metrics are used. Thus, how well each feature is weighted becomes very important as stated by Tejada and colleagues (2002). The weighting function takes the candidate cluster set and the latest record to be clustered as input. The latest record can also be a cluster since system also does merges. We have experimented with many different weighting schemes such as TF*IDF, and they will be explained in Section 8.3.1. Once the weights of each feature vector are obtained from both the new record and the candidate cluster set, the distances between candidate clusters and the new record are calculated. For calculating the distance between cluster pairs, we have experimented with many distance

algorithms and they will be explained in Section 8.3.1. When calculating the distance, we do not simply choose the minimum distance having pair like employed in the standard HAC. We employ different matching functions, and the power of our proposed system comes from these matching functions. Like stated earlier, the success of the clustering algorithms depend on the application domain. Thus, we exploit some of the characteristics of the E-commerce products. However, unlike the previous works, we only utilize the title of the products and the price which exists for every product.

The distance calculation process for each cluster is very expensive operation. Because of this reason, our matching functions are employed at each distance calculation process with an order, and if one of the matching fails, the operation is terminated, and the distance is not calculated. This speeds up the process significantly and improves the performance. The operation starts with composing the candidate cluster set for the new record. Then the distances between the new record and the candidate clusters are calculated. After all distances are calculated, the new record is either merged, or the new record becomes a new cluster based on an empirically predefined distance threshold. If new record is merged with an existing cluster, this newly emerged cluster is considered as a new record and checked against all of the existing clusters. This re-clustering process continues until no new cluster emerges. The simplified flowchart of the proposed system is displayed in Figure 6.2.

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

Furkan GÖZÜKARA

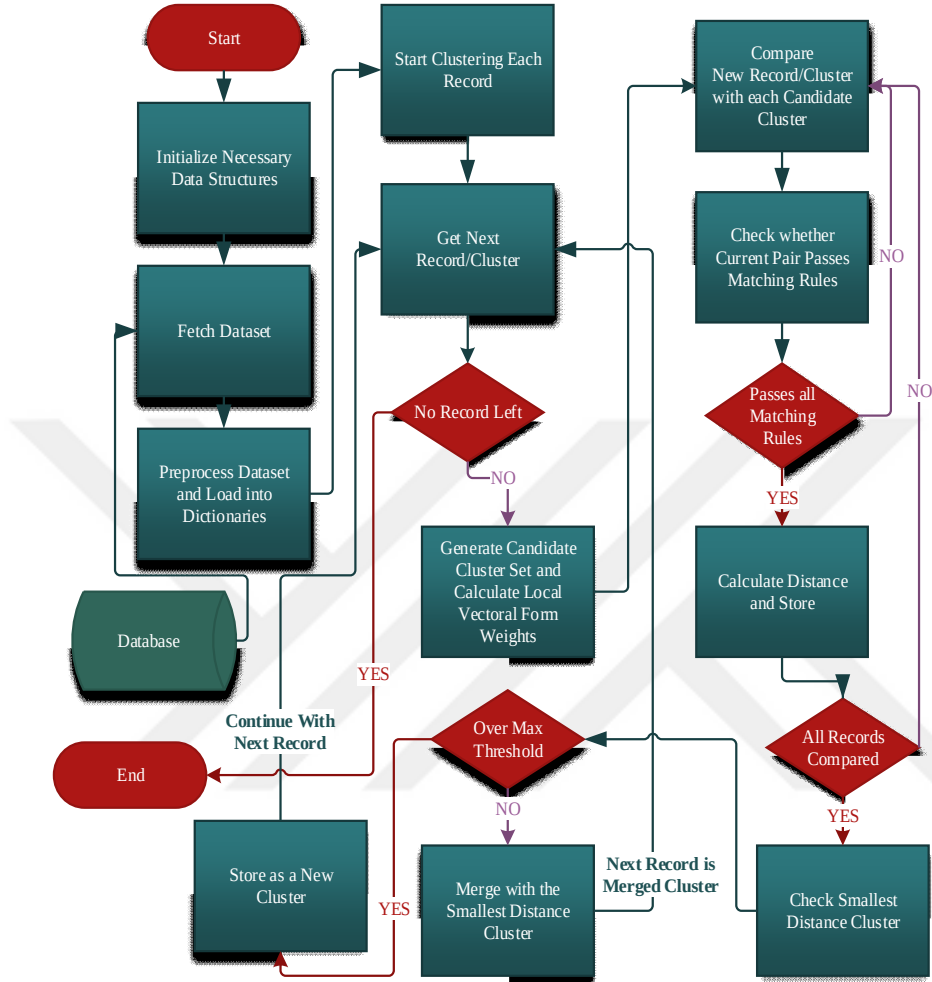


Figure 6.2. Simplified flowchart of the proposed clustering system

6.1. Matching Functions Used during Distance Computations

Matching functions are employed to speed up the distance calculation and prevent miss-merges. Thus, improve the performance of the system. We have four different matching systems, and if the compared product pairs pass all of the matching, their distance is calculated. The main difference of our approach than previous works are, previous works heavily utilizes harder to obtain features of the products such as their graphic card of notebooks or ram capacity of cell phones.

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

Furkan GÖZÜKARA

However, we utilize only title and the price feature of the products and they exists on every E-commerce product that will be displayed to the users. Moreover, our proposed system is more systematic and tuned to improve both clustering speed and the performance.

6.1.1. Price Feature Matching

The first employed matching function is based on the price feature of the products. To our best knowledge, this feature has not been used systematically like us before. If the price ratio between cluster pairs is over a predefined threshold, they cannot be the same product. For example, these below two products are very similar when only title feature is considered but their price ratio is 2.27, and they are different products:

- Sony Xperia Z2 Cep Telefonu: 1821 TL
- Sony Xperia M2 Dual Cep Telefonu: 799 TL

This price based matching system is also very efficient to prevent miss-clustering of the product and their accessories. For example Köpcke and colleagues (2012) have achieved to obtain very high success rates at the expensive products while failed to achieve such high success rates on the accessory products since they contain many features of the products that they are manufactured for. Their success could have been much higher if they had employed price ratio like we do. The price ratio check is extremely fast and prevents many of the expensive distance calculation processes of the different products.

6.1.2. Supervised Brand Filtering

The second matching system we employ is supervised brand filtering. The list of brands is manually composed by the domain expert. If the titles of the compared products have different brands, they cannot be the same product. For this

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

Furkan GÖZÜKARA

predicate, both products must have one of the predefined brands and these brands they contain have to be different. If only one of the compared products contains only brand or none have any brand, this filtering is not employed. For example “*Kingston*” and “*Corsair*” are different brands and the products contains them cannot be the same product. The entire brand list used in this study are: *nikon, canon, kingston, corsair, samsung, thermaltake, frisby, xerox, everest, inca, asus, airties, brother, hp, logitech, energizer, slink, sony, philips, lg, fakir, roadstar, mikado, epson, toshiba, lenovo, altus, beko, indesit, ugur, kaspersky, eset, sunny, vestel, hattrick, voit, altis, gigabyte, zotac, avira, symantec, panda, arnica, arcelik, western digital, and seagate.*

6.1.3. Word Conflict Check

The third matching system is word conflict check. The conflicting words list is manually composed by the domain expert and the entire list of conflicting words used in this study is shown in Table 6.4. There are certain strings that conflict, and if both string pair appears on the record pair list, they cannot be the same products. Currently, domain experts generate this pre-defined list. For example “*pc*” and “*ps3*” are conflicting words. One of them defines computer games while the other one defines PlayStation games. A good example of such case is shown below.

- PC DRAGON AGE INQUISITION
- PS3 DRAGON AGE INQUISITION

The titles of the two given products are extremely similar, and the only way to discriminate them is word conflict system. Another case we use word conflict system is the color difference on printer cartridges. If the compared product titles contain cartridge word and different color codes, they are discarded even if they are very similar such as the case below:

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

Furkan GÖZÜKARA

- Canon CRG-731C Toner Kartuş Mavi
- Canon CRG-731M Toner Kartuş Kırmızı

Table 6.4 A list of conflicting words

Conflicting Word First	Conflicting Word Second
pc	ps3
pc	psx3

6.1.4. Product Code Detection

The fourth matching function is based on product codes which are usually given by brands to identify products uniquely (Köpcke, et al., 2012). Köpcke and colleagues (2012) have heavily utilized this feature of products. However, our system works significantly different and efficient than their system. The biggest obstacle of their product code system is, it depends heavily on the public search engines. Today, none of the major search engines allows such automatic querying, and when there are thousands of products, their system becomes infeasible. The pseudocode of our product code matching algorithm is shown in Figure 6.3. The aim of the algorithm is to check whether both of the compared records contain a genuine product code or not. If both contain genuine product code, we check whether these product codes are the same or not. If both contains genuine product code and they are different, the system decides that even if they get the closest distance among the candidate set, they cannot be the same product. For example, the algorithm can successfully determine that the below products are different:

- 18.5 PHILIPS **193V5LSB262** 5MS SİYAH LED MONİTÖR
- 19.5 PHILIPS **203V5LSB26** 5MS SİYAH LED MONİTÖR

The code detection system utilizes five different parameters and one pre-defined list. These parameters and the pre-defined list are generated based on the price feature having products dataset that is explained in Section 6.2.

- (i) **min Lenght**: After conducting through experiments, we have discovered that words which have lengths that are less than or equal to three characters are usually not product codes. Thus, for a word to be a product code, we define a minimum length threshold.
- (ii) **special CodeIgnoreFeaturesList**: This list is used to hold special product features which are usually incorrectly identified as a product code. For example, 20HZ and 24INC are not product codes. To prevent miss-detection of the product code; this list holds such identifiers like “hz”, “inc”, etc.
- (iii) **minSubstringRatio**: If compared candidate products start with the certain number of common characters, they are likely to be the same product. This parameter defines the minimum percentage of the required same character count. If they do not start with the certain number of same characters, the first stage of the algorithm is terminated. Then, the second stage of the algorithm starts and second stage evaluates whether they contain the same product code or not.
- (iv) **editDistanceProductCode**: This parameter is used in two different places. First, it determines the maximum length difference of candidate codes. If length of one of the candidate code is much higher than the second one, they are discarded and evaluated at the second stage of the algorithm. This parameter also defines the maximum edit distance allowed between compared candidate product codes. If these candidates’ edit distance is greater than the pre-defined threshold, they cannot be the same product code.

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

Furkan GÖZÜKARA

- (v) **Min LC S Ratio Word Lenght**: This parameter is utilized in the second stage of the algorithm. If in the first stage of the algorithm, no genuine product codes are found, the algorithm executes the second stage to analyze further whether there are indeed genuine product codes or not. This parameter defines the minimum length of the compared candidate product code pairs. If one of the candidates has smaller number of characters than this parameter, they are not further evaluated.
- (vi) **MinLongest Common Ratio**: This parameter defines a minimum value for percentage of how many characters need to be common between the compared candidate product code pairs. The percentage is multiplied by the length of the one of the compared product codes having smaller length value.

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

Furkan GÖZÜKARA

```

1. //inputs (tokenList1 and tokenList2) are lists of tokens of the compared records
2. //output is whether compared records are different products or not
3. for each token  $t_1$  in tokenList1
4.     if ((length( $t_1$ ) > minThreshold) && ([a-Z] ∈  $t_1$ ) && ([0-9] ∈  $t_1$ ) && ({social_words} ∉  $t_1$ ))
5.         insert  $t_1$  to candidateList1;
6. for each token  $t_2$  in tokenList2
7.     if ((length( $t_2$ ) > minThreshold) && ([a-Z] ∈  $t_2$ ) && ([0-9] ∈  $t_2$ ) && ({social_words} ∉  $t_2$ ))
8.         insert  $t_2$  into candidateList2;
9. boolean ProductCodeFound = false; number MatchCount = 0; boolean ignoredBySerialNumber = false
10. for each token  $t_1$  in candidateList1
11.     for each token  $t_2$  in candidateList2 {
12.         if (length( $t_1$ ) - length( $t_2$ ) <= lengthThreshold)
13.             minLenght = min (  $t_1$ ,  $t_2$  );
14.             minPairingLenght = floor (minLenght * substringRatio%);
15.             if ( $t_1$  startswith substring of ( $t_2$ , minPairingLenght)) //e.g. g850 vs g850f
16.                 firstMod = substring of ( $t_1$ , minPairingLenght); //e.g. g850 > 0
17.                 secondMod = substring of ( $t_2$ , minPairingLenght); //e.g. g850f > 0f
18.                 if (firstMod|lastIndex| is a letter) //e.g. 0f
19.                     firstMod = substring of (firstMod, lenght(firstMod) - 1); //e.g. 0f > 0
20.                 if (secondMod|lastIndex| is a letter)
21.                     secondMod = substring of (secondMod, lenght(secondMod) - 1);
22.                 if (firstMod is a number and secondMod is a number and firstMod != secondMod)
23.                     ignoredBySerialNumber = true; //e.g. slx301 vs slx241
24.                 ProductCodeFound = true;
25.                 editDistance = Levenshtein_Distance( $t_1$ ,  $t_2$ );
26.                 if (editDistanceThreshold > editDistance && ignoredBySerialNumber == false)
27.                     MatchCount++;
28.                 else if ( $t_1$  contains  $t_2$  or  $t_2$  contains  $t_1$ )
29.                     MatchCount++;
30.
31. //Second Stage
32.     if (ProductCodeFound == false)
33.         longestCommonString = LongestCommonSubstring( $t_1$ ,  $t_2$ );
34.         minLongestLenght = minLenght( $t_1$ ,  $t_2$ );
35.         if (minLenghtThreshold <= minLongestLenght)
36.             minLongestLenght = minLongestLenght * minLongestCommonRatio;
37.         if (longestCommonString >= minLongestLenght)
38.             ProductCodeFound = true; }
39.
40. //Final Stage
41. if (MatchCount == 0 && ProductCodeFound == true)
42.     return false;
43. return true;

```

Figure 6.3. Pseudo code of Product Code Matching Algorithm

As shown in Figure 6.3, in the first part of the algorithm candidate product codes are determined and then their validity is checked. If both records found to have valid product codes and if they are different, the algorithm returns false and the clustering algorithm stops further computations for the compared products. If product code is not found in the first part of the algorithm, the second stage of the algorithm is executed and further evaluation of product code detection is made. In

the final evaluation, it is checked whether both of the records contains product codes or not. If both records contains genuine product codes and if they are same, algorithm returns true and if none of the found codes are not the same, algorithm returns false.

6.2. Dataset of Product Record Linkage

To develop a dataset to make performance evaluation of the proposed clustering method, top 50 E-commerce websites in Turkey are crawled by our focused E-commerce crawler software, and title and price feature of the products along with additional features are saved. In the record linkage part of this thesis study, only title and price feature of the products are utilized. These two features always exist in a commercial product. If the price feature is not available, it means that the product is either out of stocks or expired thus, they should not be displayed to the consumers in CSA systems. So, such products can be safely ignored. 763,886 product offers are obtained from the crawling process. From these offers, 250 different offers are randomly picked to compose the dataset. The products are selected only from Vatanbilgisayar.com. Thus, every product is ensured to be different. Because usually, an E-commerce website does not list the same product more than once. The only exception we have found is, some websites tend to list different colors of the same product as different, and this issue is taken into consideration when manual labeling is done. Then the entire dataset is manually scanned to find which one of the other E-commerce websites also contains these products and then the same products from the other websites are labeled as the same product. When we decide which products are the same, different colors of the same products are taken as the same product except for the products such as toner cartridges where color is important and differentiates the products. There are 250 different, 1800 (from 49 E-commerce websites) total, and 1721 (from 48 E-commerce websites) price feature having products in the final composed dataset. The dataset of the products which have price feature is called as DSE for easier

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE

Furkan GÖZÜKARA

reading in the rest of the study. Additionally, we have composed macro-average dataset which contains 100 different sets for a more fair evaluation. For composing the macro-average dataset, 100 different sets are composed by selecting 1,000 random records that are randomly sorted from the price having records. Each set in the macro-average dataset contains only unique records.

We have published these datasets on a GitHub⁴⁰ repository with different formats including the preprocessed and the used format in our research. As far as to our knowledge, this is the first ever E-commerce products dataset published in the literature which contains data from more than two sources and over 90 different categories of products. More details about the dataset can be found in the GitHub repository. We also have to note that, the order of the input does matter when incremental HAC is made. Thus, macro-average dataset is important to evaluate the success of the algorithms correctly. Moreover, to ensure comparability between different algorithms and methods, we have used the same order of the records in all of the experiments.

⁴⁰ <https://github.com/FurkanGozukara/Record-Linkage>

6. PROPOSED CLUSTERING SYSTEM for PRODUCT RECORD LINKAGE
Furkan GÖZÜKARA



7. PROPOSED SEARCH ENGINE and USER INTERFACE

An internal search engine is necessary to translate user input and retrieve the related data from the database. Once the data is retrieved, it is processed and displayed to the user in an elegant way. How the results are displayed to the users is the user interface. Both of the tasks are very important. However, the success of the internal search engine system solely depends on the success of the record linkage and the sentiment analysis. The reason is, internal search engine depends on how well the clusters are composed and how correctly the user comments are classified.

7.1. Search Engine

The main aim of the internal search engine is translating user query into meaningful features and then scanning the database for the related products. We propose two different internal search engine systems. The first system is used to search for the most related products according to the user input. The second system is employed to look for the most popular products in the user provided category label.

7.1.1. Most Related Products

Finding most related products is a three stage process. In the first stage, the record linkage system clusters the crawled products. In the second stage, the titles of these products are processed and saved in the database. In the second stage, we propose the following system:

- i. The title is preprocessed and split into words by space character.
- ii. When preprocessing is being made, a new technique *merge possible product names* is employed. In this technique, if a word which does not have any numerical value is followed by an integer value, the merged

7. PROPOSED SEARCH ENGINE and USER INTERFACE

Furkan GÖZÜKARA

version of this word pair appended to the end of the product title. For example if the title is “*GTX 780*”, it becomes “*GTX 780 GTX780*”. This process eliminates the different typing schemes of E-commerce websites. This process is not necessary during the record linkage task since 2-Gram transformation is employed.

- iii. Once the title of the product is preprocessed and split into words, each word of the product is saved in the database with the unique product key. Unique product keys are automatically given during the crawling process to each crawled product page. Moreover, proper indexes are defined for this table for very fast look ups.

The third stage is where the user input is taken and database is queried. For the third stage we propose the following system:

- i. User query is preprocessed and split into words. Before the user query is split into words, each word is merged with the following word and appended to the end of the query. The reason is, some products are written merged and split by the E-commerce websites. For example “*GTX 780*” or “*GTX780*”. So this process improves the recall rate of the database scan.
- ii. Each word is queried in the database and products that contain these queried words are fetched.
- iii. Cluster table is queried with these products and clusters that contain these products are fetched.
- iv. Cluster table is queried with the obtained cluster IDs and which products belongs to these clusters are fetched.
- v. The necessary information of all obtained products are fetched from the database.
- vi. Product information are processed.

7. PROPOSED SEARCH ENGINE and USER INTERFACE

Furkan GÖZÜKARA

- vii. Distance of the each product against the user query is calculated with the same system employed in record linkage. While the distance is calculated, the extra *word merge* applied to the user query is not employed. Word merge is employed only when scanning the database.
- viii. Once all of the distances are calculated, they are sorted by ascending order and each cluster information is composed. So the order of the products that are presented to the user is, from most related to lesser related. For most related products search system, not information of all of the products but the information of the clusters they belong are presented. Thus, user can choose which product they are looking for and see the details.

The key element of this system is, not only the products that precisely contain user query terms, but all of the products that belongs to same cluster that user query term containing products are fetched. This ensures that different naming schemes or missing attributes from the product titles are handled correctly. The examples with the user interface are presented in Section 7.2.

7.1.2. Most Popular Products

The idea of most popular products is based on the finding popular products in a given category. For example a user may want to purchase a printer but may not be sure about which product he/she should buy. In this case, the most popular products search system can help user to determine which product he/she should purchase.

Most popular products system is composed of three stages. In the first stage, which is during the crawling process, each category of the products are saved with parent-children hierarchy. This process is the task of the proposed crawler system. In the second stage, the user comments are classified into negative or positive classes by the proposed sentiment analysis system. The third stage is

7. PROPOSED SEARCH ENGINE and USER INTERFACE

Furkan GÖZÜKARA

where the user query is translated and most popular products are retrieved. We propose the following system for this task:

- i. The product categories that matches with the user query are fetched. In this process, we do not apply any preprocessing because category tables are case and accent insensitive. For example if a product category is “Ekran Kartı” and if the user query is “ekran kartı”, they will match.
- ii. The sub-categories which are children of the categories that are fetched in the first step, are found.
- iii. All of the products that belong to the found categories are retrieved.
- iv. All of the clusters that contains these products are fetched.
- v. All of the products that belongs to these clusters are found.
- vi. All of the comments that belong to these products are retrieved.
- vii. The popularity scores of all of the clusters are calculated.
- viii. The clusters are presented to the user with the popularity score in descending order.

For calculating the popularity score, we utilize the following formula that we propose where $score_{P_i}$ is the popularity score of the i -th cluster, C_{p_i} is the total number positive product comments in the i -th cluster, C_{n_i} is the total number negative product comments in the i -th cluster, S_i is the number of different E-commerce stores in the i -th cluster. Several other methods such as (Zhang & Tran, 2008; Baccianella, et al., 2009; McCallum, et al., 2000; Zhang, et al., 2011) are proposed to rank products according to customer reviews. However, to our best knowledge, our scoring system has not been proposed previously. Further analysis on such ranking systems may be done as future work.

$$scoreP_i = (C_{pi} - C_{ni}) \times S_i$$

There are several key components of the above proposed system. Selection of the sub-categories is important to present user all of the products in that particularly searched category. Another key issue is selection of the clusters to fetch all products that belongs to these clusters. This eliminates different category naming problem. Many E-commerce websites tend to use different category hierarchy, or different category naming schemes. Our proposed system ensures that even if only a single product have the user searched category, all other products that share the same cluster are retrieved. Relevant examples are provided in Section 7.2.

7.2. Proposed User Interface

The user interface should be elegant and easy to understand. We have designed user interface by using Twitter Bootstrap⁴¹ 3.3.6 framework. Bootstrap framework provides both easier development and fully mobile friendly user experience. The user can search either best related products or most popular products in a given category in the main user interface. The user interface screen is shown in Figure 7.1.



Figure 7.1. Product search engine user interface

⁴¹ <http://getbootstrap.com/>

7. PROPOSED SEARCH ENGINE and USER INTERFACE

Furkan GÖZÜKARA

When a user search for a product, the most related results are displayed as shown in Figure 7.2 where green up arrow display number of positive comments and red down arrow display number of negative comments. Number of offers and number of comments are cumulative values in the cluster. The title of the cluster displayed to the user is determined based on the similarity of the product to the user's query. In each cluster, products are ordered by their similarity to the search query and title of product having the highest similarity is chosen to be shown to the user as the title of the cluster.

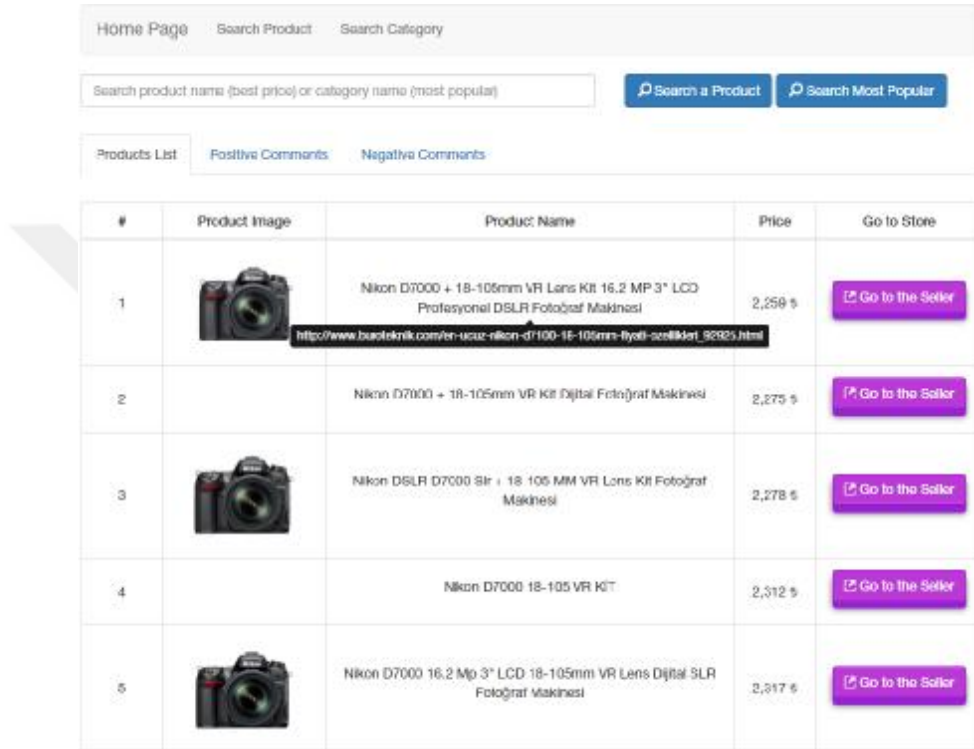
Home Page Search Product Search Category						
fotoğraf			Search a Product Search Most Popular			
#	Product Image	Product Name	Price		# of Offers	# of Comments
1		NIKON D7000 18-105MM VR KIT DİJİTAL FOTOĞRAF MAKİNESİ	Min 2,250 ₺	Max 3,106 ₺	16	↑ 45 ↓ 6
2		Nikon D510 Body DSLR Dijital Fotoğraf Makinesi	Min 6,999 ₺	Max 9,899 ₺	5	↑ 0 ↓ 0
3		Nikon CoolPix S2800 Dijital Fotoğraf Makinesi	Min 269 ₺	Max 297 ₺	7	↑ 1 ↓ 0
4		Nikon CoolPix P510 Dijital Fotoğraf Makinesi	Min 1,206 ₺	Max 1,206 ₺	1	↑ 0 ↓ 0
5		Nikon Coolpix S800C Siyah Dijital Fotoğraf Makinesi	Min 549 ₺	Max 689 ₺	3	↑ 0 ↓ 0

Figure 7.2. Screenshot of Search a Product results

7. PROPOSED SEARCH ENGINE and USER INTERFACE

Furkan GÖZÜKARA

The user then can click the name of the product that they are interested in and see the details of that product as shown in Figure 7.3. As shown in Figure 7.3, if user hovers their mouse on the product title, it shows the URL where it is sold.



#	Product Image	Product Name	Price	Go to Store
1		Nikon D7000 + 18-105mm VR Lens Kit 16.2 MP 3" LCD Profesyonel DSLR Fotoğraf Makinesi http://www.buroluk.com/en-ucuz-nikon-d7000-18-105mm-lyali-sesliket_92925.html	2,259 ₺	Go to the Seller
2		Nikon D7000 + 18-105mm VR Kit Dijital Fotoğraf Makinesi	2,275 ₺	Go to the Seller
3		Nikon DSLR D7000 Sır + 18-105 MM VR Lens Kit Fotoğraf Makinesi	2,278 ₺	Go to the Seller
4		Nikon D7000 18-105 VR KIT	2,312 ₺	Go to the Seller
5		Nikon D7000 16.2 Mp 3" LCD 18-105mm VR Lens Dijital SLR Fotoğraf Makinesi	2,317 ₺	Go to the Seller

Figure 7.3. Screenshot of how details of a product are presented to the users

User can also see positive and negative comments made for that product as shown in Figure 7.4. These comments are cumulatively collected from all of the E-commerce stores that particular product is sold. As shown in Figure 7.4, some comments do have title and some do not have. Moreover, some comments have positive and negative parts and they are also accordingly displayed. These differences come from different comment systems of the E-commerce websites.

7. PROPOSED SEARCH ENGINE and USER INTERFACE

Furkan GÖZÜKARA

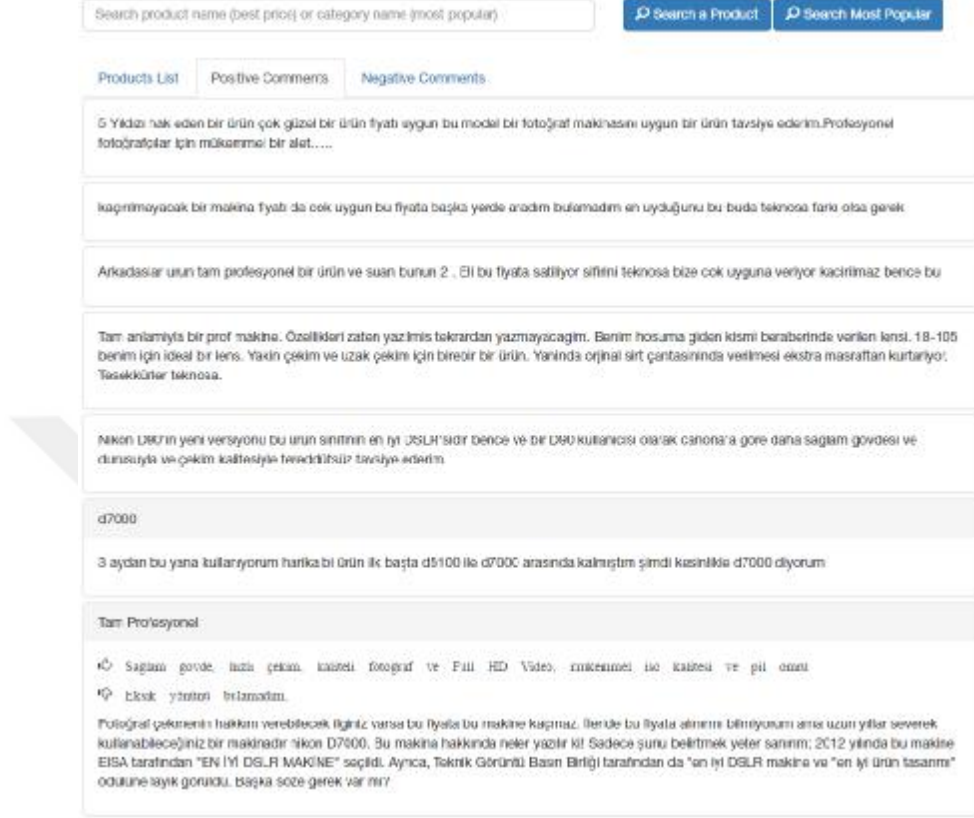


Figure 7.4. Screenshot of how positive comments of a product are displayed

Users can also search for most popular products in a particular product category as shown in Figure 7.5. The title of the cluster displayed to the users is determined based on the price of the product. In each cluster, products are ordered by their price and title of the lowest price having product is chosen to be shown to the users as the title of the cluster. However, deciding which product's title is best to show can be a future research topic.

7. PROPOSED SEARCH ENGINE and USER INTERFACE

Furkan GÖZÜKARA

Home Page Search Product Search Category						
ekran kartı			Search a Product		Search Most Popular	
#	Product Image	Product Name	Price		# of Offers	# of Comments
1		SAPPHIRE 4GB R9 290X OC TRI-X GDDR5 512 Bit 11226-00-40G HDMI 2xDP1 ATI 16X	Min	Max	10	 3  1
			1,144 ₺	1,892 ₺		
2		Asus GTX780 ROG POSEIDON GDDR5 3GB 384Bit Nvidia GeForce DX11.2 Ekran Kartı	Min	Max	7	 3  1
			1,225 ₺	1,396 ₺		
3		EVGA GTX980 SuperClock ACX 4GB GDDR5 256 Bit Ekran Kartı - 04G-P4-2983-KR	Min	Max	3	 2  1
			2,055 ₺	2,128 ₺		

Figure 7.5. Screenshot of Search Most Popular results

7. PROPOSED SEARCH ENGINE and USER INTERFACE

Furkan GÖZÜKARA



8. EXPERIMENTAL EVALUATION

8.1. Focused Web Crawler Development Experiments

In this section experiments performed and their results are presented to evaluate the efficiency and the performance of the proposed methods. All of the experiments are performed on a personal home computer having the below configurations:

- **Operating System:** Windows 8.1 (6.3) Enterprise Edition 64-bit (Build 9600)
- **CPU:** Intel Core i7-2600K CPU @ 3.40GHz, 1 CPU - 4 Cores - 8 Threads, Frequency 4544.05 MHz (45 * 100.98 MHz) (Information received by CPU-Z⁴² 1.72 x64)
- **RAM:** 32768 MB, DDR3-1346 - Dual Channel, Frequency 673.2 MHz (1:5), Timings 9-9-9-24 (Information received by CPU-Z 1.72 x64)
- **Hard Disk:** 2x OCZ Vertex 4, Raid 0 setup with stripe size 64 KB, Write-Cache Buffer Flushing: Enabled, Cache mode: Write back, Sequential read 742 MB/s, write 772 MB/s, Random 512 KB read 482 MB/s, write 486 MB/s. Tests are done at CrystalDiskMark⁴³ 3.0.4 x64. Access time: 0.127 milliseconds (access time tested by HD Tune Pro⁴⁴ 5.50). Other than from synthetic tests, when reading a large amount of data (30 GB+) from the SQL server, the windows task manager displayed 630 MB/s read speed with 6000 milliseconds average response time

⁴² CPU-Z is a freeware that gathers information on some of the main devices of your system, <http://www.cpuid.com/softwares/cpu-z.html>

⁴³ CrystalDiskInfo is a HDD/SSD utility software (open source) which supports S.M.A.R.T and a part of USB-HDD, <http://sourceforge.jp/projects/crystaldiskinfo/>

⁴⁴ <http://www.hdtune.com>

- **Internet Connection (Network Speed):** 50 Mbit/s download, 5 Mbit/s upload, personal home fiber connection, provider: Turkcell Superonline⁴⁵
- **Best Performance Settings:** HTTP compression is enabled, initialization of the thread pool is enabled, garbage collector mode is set to *server*, and concurrent connections limit is increased.

All of the tests (except source code compression) are executed for 30 minutes and in each minute statistics of the conducted tests are collected. In order to collect network traffic statistics, Fiddler⁴⁶ framework is used. The RAM usage and the CPU usage statistics are collected by using *PerformanceCounter* class and 1-second polling methodology which are also used by Windows Task Manager. We use a measure which we call as “UI responsiveness” to evaluate the performance of our system. The main task of our system is called as UI and therefore UI responsiveness is the response rate of the UI to the UI changes. UI responsiveness is a direct indication of the system performance when all of the tasks are started independent from the UI thread since UI may lag only when the application itself is throttling in such scenario. The UI responsiveness is measured by the following strategy: (i) only a class is used to refresh UI by *Monitor.TryEnter* locking strategy as shown in Appendix Code Snippet 5; (ii) after each UI refresh is completed, the time of the refresh is recorded; (iii) the UI is updated per 0.5 seconds. We have chosen updating per 0.5 seconds because “*high update speed*” profile of Windows Task Manager also uses 0.5 seconds interval to refresh the statistics. So after the application is started, each second has exactly 2 times UI refresh process. If the corresponding second has any missing refresh, it is counted as UI freeze, if the second has more than 2 times UI refresh, it is counted as UI lag (delayed update consecutively). The total responsiveness of the UI is measured as

⁴⁵ <http://www.superonline.net/kesfet/fiber-internet>

⁴⁶ The free web debugging proxy for any browser, system or platform, <http://www.telerik.com/download/fiddler>

the number of correct UI refresh matches in each second divided by the number of total expected UI refresh. So if the UI is refreshed 2 times within each second, the UI responsiveness is 100%. Refresh count over than 2 or less than 2 in a second means there is either UI lag or UI freeze which both reduce UI responsiveness.

Average application threads count is calculated by using 1-second polling methodology and `Process.GetCurrentProcess().Threads.Count`⁴⁷ method. Maximum 300 concurrent crawling agents are used for experiments and maximum crawling count for each website is limited individually. We have used 50 different E-commerce sites for the experiments and some of them are listed in Table 4.1. When we have tested the limit for the default number of connections per host, a single website is used for experimenting so that the software can possibly reach the number of concurrent connections per host threshold. Every crawled page is processed to extract URLs in the crawled page, determine the status of crawled page as canonical or a product page or not, etc. However, page processing that is applied to product pages involves extraction of product title, product categories, product features, product comments, etc. To determine type of the page that is processed, EcCrawler needs to determine whether the crawled pages are “a product page” or “not” and this is decided at the link extraction phase by applying manually generated rules. If the page is not a product page, it is not saved for further processing. During the tests, the number of page processing tasks’ count was set to 100.

The tests, other than the source code compression, depend on the network capacity, the hard disk performance, and the SQL-Server caching. Because of these reasons, all of these experiments are conducted after 1 am local time for politeness and more objectivity. In addition, before the test, the database is reset (returned to initial values), the computer’s IP address is changed and the computer is restarted. These actions were taken in order to solve the following possible issues:

⁴⁷ MSDN, [https://msdn.microsoft.com/en-us/library/system.diagnostics.process.getcurrentprocess\(v=vs.90\).aspx](https://msdn.microsoft.com/en-us/library/system.diagnostics.process.getcurrentprocess(v=vs.90).aspx)

- It can be expected that the E-commerce websites' server load will be lower after the midnight due to decreased visitors' activity. After midnight crawling also improves the chances of not harassing the crawled websites.
- Both home's internet connection network load and the crawled E-commerce websites' server network load changes throughout the day. Because of this reason, making all such network load based tests after 1 AM local time should improve the reliability of the experiments.
- Windows 8.1's disk system, SQL-Server, and Windows 8.1's DNS system make heavy caching of many different requests to improve the performance. Conducting experiments consequently can yield unfairness among the various tests because of these caching. After each one of the tests, the computer is restarted to conduct the tests in equal conditions.
- Since the home network is used to carry out the tests and the ISP provides dynamic IP address, the IP address is changed after each one of the tests to prevent any possible IP blockage. However, this may not have been necessary because while conducting the experiments, we did not notice any IP blockage or we did not receive any complaint from any of the crawled web servers' administrators. Not receiving any complaints or not getting blocked may show that the methods used are sufficient to handle multiple connections from each client without any trouble.

8.1.1. Database Communication

In order to verify the effectiveness of the proposed database connection system, we have conducted the following 3 different tests: (i) in the first test scheme, no batch database access is used, and for each database access a new connection session is opened and closed; (ii) in the second test scheme, batch database access methodology is used and for multiple database accesses, a single connection session is used and closed; (iii) in the third test scheme, a static

connection is employed for all of the database accesses between all of the different threads by using locking methodology to ensure thread safety, and it is never closed. The difference from the batch database access mode is, it uses a single connection to execute multiple different queries. So it should not be confused with batch SQL queries. Therefore, it reduces overhead of SQL-Server connection session handling.

Database communication experiments are conducted on the EcCrawler system. However, we did not observe any significant difference between these three modes. The reasons for this observation may be that the system has a lot of very heavy system demanding tasks when compared to the database session generation and there may be not enough number of database connection procedures to observe the effectiveness of the proposed systems. Additionally, the executed queries in the EcCrawler were already heavily tuned to obtain maximum performance. Because of these reasons, we decided to conduct synthetic tests that will heavily use the database connections.

In order to carry out the synthetic tests, a new table is designed which have a single *Integer 32(int)* primary key column. The numbers between 1 and 10,000 are inserted into this table as primary keys. Then a new software is developed to execute the tests. This software spawned 250 *Tasks* (each one is separate from the main thread) and each one of these *Tasks* continues to work constantly until the application is terminated. These *Tasks* queries the database and selects a single *DataRow* from the synthetic test table by generating a random primary key predicate. Then database accesses of the *Tasks* are counted cumulatively. In the batch database access methodology, a single new database connection for each *Task* is used 50 times before it is closed. The table and a sample query used in the experiments are presented in Figure 8.1. The reason for choosing such a high number of *Tasks* is to maximize the system resources usage in all cases. These values are not optimal values for the best system performance. Also, very simple database design is chosen to evaluate the database connection performance rather

than the database querying performance. The results of these experiments are shown in Table 8.1.

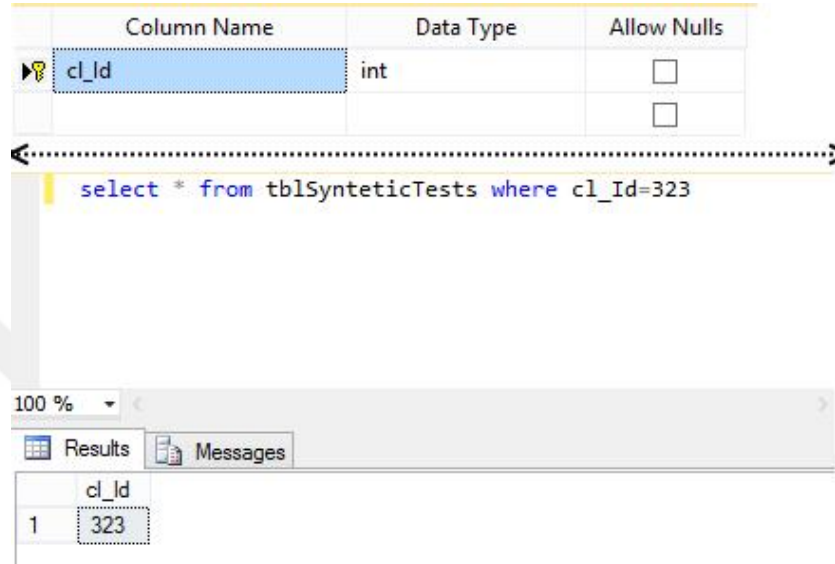


Figure 8.1. Table and query used in the synthetic tests

Table 8.1. Experimental results for synthetic database connection

Test Scheme	Average CPU Usage of the Test Software	Average CPU Usage of the SQL-Server	Average Number of Completed Queries per Minute
Scheme 1 - Single Query per Database Connection	53.90%	45.12%	3,120,492
Scheme 2 - Batch Mode – Multiple Queries per Database Connection	51.21%	45.65%	3,982,170
Scheme 3 - A Single Static Connection for all of the Threads and the Queries	99.17%	0.49%	58,891

When the results in Table 8.1 are analyzed, Scheme 2 is the best performer by almost equalizing the CPU utilization of both software and SQL-Server. Thus, it also achieves to obtain highest number of queries per minute. When Scheme 1 and Scheme 2 are compared, the CPU usage of software in Scheme 1 is slightly higher than that of the Scheme 2. The possible reason of this can be that the software spends more time to establish a new session for each query instead of using the same session for multiple queries. When the results of Scheme 3 are interpreted, we see that the test software uses a massive amount of the available system resources, and SQL-Server utilization is poor. The possible reason of this is, each time a Task demands a database access, it has to lock the static database connection. If the connection is already locked, it has to wait. This locking process is significantly system demanding, and this is probably the main cause of unbalanced CPU usage between the test software and the SQL-Server and results in a lower performance.

8.1.2. Source Code Compression

Page processing is a much more challenging task when it is compared to crawling pages and saving their source code task. Because of this reason, page processing and page source saving tasks have to be asynchronous. Since the crawling task is much faster than the page processing task, to achieve asynchronous execution, source codes of the web pages have to be kept until they are processed. However, storing the raw source code of the crawled pages takes huge space both on the disk and in the RAM. Finding the optimal compression algorithm for the developed crawler can help to reduce space requirement. If the disk space usage is more crucial than the CPU requirement for the developed software, a more aggressive compression can be chosen. However, the obtained experimental results indicate that one of the compression methods clearly surpasses every other method by the *CPU_requirement/compression* ratio.

For the compression tests, the dataset which is generated by EcCrawler is used. 100,000 pages are randomly chosen from the dataset. The features of the pages in the dataset are as follows: maximum source code size (UTF-8) of a single page is 8.54 Megabytes, the average size is 86.68 Kilobytes, the minimum size is 23.68 Kilobytes, and total size of all uncompressed source codes size is 8.26 Gigabytes. When testing Gzip compression algorithm, .NET framework is used because it natively supports Gzip. For testing the other compression algorithms, 7-Zip⁴⁸ library is used with `CompressionLevel` set to `Normal`. To use 7-Zip DLL file in C#, `SevenZipSharp`⁴⁹ is used as a wrapper.

The results of the compression experiments are presented in Table 8.2 where **Min**, **Max**, and **Avg Compressed Source Size** are the minimum, the maximum, and average size of the compressed source code of a crawled page in kilobytes; **Min**, **Max**, and **Avg Compression Time** are the minimum, the maximum, and average elapsed time for compressing source code of a crawled page in milliseconds; **Min**, **Max**, and **Avg Decompression Time** are the minimum, the maximum, and average elapsed time for decompressing source code of a crawled page in milliseconds; **Min**, **Max**, and **Avg Compression Benefit** are the minimum, the maximum, and average of the percentage difference between the size of the compressed and raw source codes of a crawled page; **Avg CPU Usage** is the average CPU usage of the system while compression experiments are running.

⁴⁸ Open source 7-Zip is a file archiver with a high compression ratio, <http://www.7-zip.org>

⁴⁹ Managed 7-zip library written in C#, <https://sevenzipsharp.codeplex.com/>

Table 8.2. Comparison of the compression methods

Mode:	Gzip	BZip2	Deflate	Deflate 64	Lzma	Lzma2	Ppmd
Min Compressed Source Size (KB):	9.01	9.30	8.75	8.75	8.74	8.75	8.08
Max Compressed Source Size (KB):	8,471	8,458	8,472	8,472	8,470	8,472	8,577
Avg Compressed Size (KB):	24.97	24.09	22.78	22.78	22.78	22.79	21.80
Min Compression Time (milliseconds):	0	13	27	15	22	27	13
Max Compression Time (milliseconds):	366	637	4,700	1,977	4,679	4,718	3,930
Avg Compression Time (milliseconds):	1.53	42.03	61.27	39.20	57.99	62.07	38.37
Min Decompression Time (milliseconds):	0	10	20	9	19	20	11
Max Decompression Time (milliseconds):	178	732	541	527	567	538	4,369

Table 8.2. Comparison of the compression methods (Continued)

Avg Decompression Time (milliseconds):	0.39	31.14	57.93	30.57	57.49	57.15	30.66
Min Compression Benefit (%):	2.42	2.40	3.14	3.14	3.16	3.14	1.94
Max Compression Benefit (%):	91.77	93.33	92.93	92.93	92.93	92.93	93.50
Avg Compression Benefit (%):	71.18	72.20	73.70	73.70	73.71	73.70	74.84
Avg CPU Usage (%):	17.51	16.67	16.22	19.16	16.26	16.43	17.31

In Table 8.2, the best values for each performance evaluation parameter are written in bold face. According to the results that are presented in Table 8.2, Gzip clearly outperforms the other methods when both the compression ratio and the compression times are considered. Even though the Gzip obtains lower average compression ratio than all of the other methods, it is at least 15 times faster than the other methods for the compression task and 78 times quicker than the other methods for the decompression task. The compression performance gain of the other methods is extremely low when compared with their increased time requirement and because of this reason EcCrawler uses Gzip.

8.1.3. Using Compression While Fetching Pages

Today, the majority of the web servers supports compressed data transferring, however by default, .NET does not make the HTTP requests with compression enabled. Necessary headers are needed to be included in the requests, and automatic decompression needs to be enabled as shown in Appendix Code

Snippet 8, to fetch web pages with HTTP compression. The results of the HTTP compression enabled tests, and the default `HttpWebRequest` class tests are presented in Table 8.3 where the gray highlighted rows are the best performance settings (i.e., compression enabled) and the not highlighted (white) rows are the results of the same settings when HTTP compression is deactivated.

In Table 8.3; **Crawl Success** is the number of successfully crawled web pages; **Crawl Fail** is the number of unsuccessfully crawled web pages; **Avg # of Pages per Minute Crawl** is the average number of successfully crawled web pages per minute; **Avg # of Processed Pages per Minute** is the average number of processed pages per minute; **Avg Download Speed per Second** is the average bandwidth in kilobytes used by the application per second to download web pages; **Avg Upload Speed per Second** is the average bandwidth in kilobytes used by the application per second to upload data (upload is used for only connecting the websites) to the web servers; **Discovered Unique URLs** is the number of discovered unique web pages. Other metrics used in the table are self-explanatory.

Table 8.3. The effect of compression enabled HTTP requests experiment

HTTP Compression (Minutes)	5	10	15	20	25	30
Crawl Success	30,917	62,595	98,092	134,875	171,821	210,189
	19,860	40,824	62,178	84,260	106,032	126,735
Crawl Fail	114	459	1,348	1,622	1,697	1,754
	199	486	740	991	1,828	2,818
Avg # of Pages per Minute Crawl	6,086	6,195	6,485	6,693	6,827	6,960
	3,914	4,046	4,115	4,184	4,213	4,198
Avg # of Processed Pages per Minute	815	805	780	738	715	704
	701	722	699	693	679	670
Avg Download Speed Per Second (KiloBytes)	2,536	2,550	2,665	2,633	2,631	2,660
	5,645	5,759	5,790	5,769	5,791	5,805
Avg Upload Speed Per Second (KiloBytes)	38	39	42	44	44	45
	21	21	21	22	22	22
Discovered Unique URLs	153,994	207,914	259,737	309,515	346,910	383,869
	115,564	168,024	211,293	236,519	261,804	291,895
Avg RAM Usage (MegaBytes)	1,003	1,106	1,177	1,229	1,275	1,292
	1,031	1,118	1,153	1,179	1,198	1,215
Avg CPU Usage (%)	59.74	62.07	63.36	63.89	64.35	64.71
	32.78	32.45	33.01	33.01	33.11	33.04
Avg # of Threads Spawned	542	555	553	554	558	562
	623	644	663	696	725	756
UI Responsiveness (%)	98	98.41	98.22	97.95	97.83	97.80
	98.5	98.58	98.66	98.62	98.56	98.61
UI Freeze (%)	2	1.58	1.77	2.04	2.81	2.19
	1.5	1.41	1.33	1.37	2.16	1.38
UI Lag (%)	0.33	0.16	0.11	0.16	0.2	0.16
	0	0.08	0.05	0.08	0.06	0.05

When Table 8.3 is analyzed the benefit of HTTP compression is clearly observed. As mentioned in the testing platform specifications, the maximum download speed of the internet connection was 50 Mbits per second, which is roughly equal to 6400 KiloBytes per second. Once the HTTP compression was disabled, the testing system network load was maximized, and this hindered the number of pages fetched per minute. Enabling HTTP compression increases the number of pages crawled per minute by 65.79% and decreases the bandwidth usage by 54.17%. Another important aspect of these tests is the number of processed pages per minute. In the first 5 minutes, the number of processed pages per minute is significantly higher than the HTTP compression disabled mode. However, this difference decreases as the time passes. The reason of this is, at the beginning, there are not enough product pages to process until the page processing tasks count is maximized. Thus, the faster page crawling mode is obtained by processing more pages. However, as more pages are crawled, the number of obtained product pages increases and the maximum capacity of the page processing tasks are surpassed and the gap is closed and the average number of pages processed per minute for both modes almost becomes equal.

The reason of having difference between the two modes in terms of average number of threads spawned is that, when the HTTP compression is enabled, it fetches pages faster. Thus, the operation is completed faster, and the crawler thread is terminated quicker than the HTTP compression is disabled mode. When the HTTP compression is enabled, the UI responsiveness reduces slightly due to the fact that, in the case of compression, more pages are processed, thus heavier processing is done.

8.1.4. Initialization of Thread Pool

By default, .NET framework spawns a certain number of threads by the pre-defined heuristics to maximize system performance. However, .NET framework heuristics may result in lower performance by using a smaller portion

of the available resources. Hence, initialization of thread pool manually may improve the system performance greatly. The initialization of the thread pool is shown in Appendix Code Snippet 1. In Table 8.4, the results of the initialization of the thread pool experiment are presented. In the table, the gray highlighted rows include results of the best performance settings in which thread pool is initialized, and the not highlighted (white) rows display results of the same settings except that the thread pool is not initialized.

Table 8.4. Result of the thread pool initialization experiment

Thread Pool (Minutes)	5	10	15	20	25	30
Crawl Success	30,917	62,595	98,092	134,875	171,821	210,189
	16,727	45,068	71,034	97,986	121,094	131,132
Crawl Fail	114	459	1,348	1,622	1,697	1,754
	110	252	547	715	969	1,182
Avg # of pages per Minute Crawl	6,086	6,195	6,485	6,693	6,827	6,960
	3,017	4,260	4,550	4,744	4,691	4,244
Avg # of Processed Pages per Minute	815	805	780	738	715	704
	377	585	678	692	701	664
Discovered Unique URLs	153,994	207,914	259,737	309,515	346,910	383,869
	99,360	155,312	195,540	228,970	252,695	263,187
Avg RAM Usage (MegaBytes)	1,003	1,106	1,177	1,229	1,275	1,292
	838	1,056	1,139	1,186	1,210	1,236
Avg CPU Usage (%)	59.74	62.07	63.36	63.89	64.35	64.71
	36.68	40.88	40.41	39.67	38.14	34.14
Avg # of Threads Spawned	542	555	553	554	558	562
	358	417	448	466	478	500
UI Responsiveness (%)	98	98.41	98.22	97.95	97.83	97.80
	41.5	58.33	64.66	68.04	69.46	67.47
UI Freeze (%)	2	1.58	1.77	2.04	2.81	2.19
	58.5	41.66	35.33	31.95	30.53	32.52
UI Lag (%)	0.33	0.16	0.11	0.16	0.2	0.16
	22.33	19.25	17.33	17.20	17.16	18.47

Thread pool experiment clearly shows the performance gain at the UI responsiveness. Even though the number of active threads increases over the time as the framework optimizes the count of the active threads, the performance loss of the default settings is still significant. Initialization of the thread pool utilizes the application for using the system resources more efficiently and achieves better performance.

8.1.5. Garbage Collector Mode Experiment

By default, the garbage collector mode is set to workstation, however, this mode is designed for the single core processors. Nowadays, the majority of the computers sold in the market have multi-core processors. We did not have any single-core computer to test the mode difference, however, on the testing platform, changing workstation mode to server mode improves overall performance significantly. The garbage collector mode is set as shown in Appendix Code Snippet 6. In Table 8.5, the gray highlighted rows present results for the best performance settings in which garbage collector mode is set to server and the not highlighted rows include results for the same settings except garbage collector mode is workstation.

If all of the tasks are running as background threads, the UI thread is not used for any processes, and if there exists enough system resources to refresh the UI, there should not be any UI responsiveness problem. However, if the application utilizes many threads and these threads heavily uses system resources as in EcCrawler, default settings may fail to supply necessary system resources to the application, and this may cause the UI to freeze or lag. We propose that this problem can be solved by manual initialization of the thread pool and setting the garbage collector mode.

Table 8.5. Results of the garbage collector mode experiment

Garbage Collector Mode (Minutes)	5	10	15	20	25	30
Crawl Success	30,917	62,595	98,092	134,875	171,821	210,189
	7,811	17,563	25,740	32,789	39,868	48,147
Crawl Fail	114	459	1,348	1,622	1,697	1,754
	29	85	104	121	131	162
Avg # of Pages per Minute Crawl	6,086	6,195	6,485	6,693	6,827	6,960
	1,529	1,730	1,690	1,616	1,568	1,580
Avg # of Processed Pages per Minute	815	805	780	738	715	704
	518	622	627	633	619	618
Avg Download Speed Per Second (KiloBytes)	2,536	2,550	2,665	2,633	2,631	2,660
	518	550	545	510	495	498
Avg Upload Speed Per Second (KiloBytes)	38	39	42	44	44	45
	9	10	10	10	10	10
Discovered Unique URLs	153,994	207,914	259,737	309,515	346,910	383,869
	70,157	102,738	126,387	139,490	149,314	163,116
Avg RAM Usage (MegaBytes)	1,003	1,106	1,177	1,229	1,275	1,292
	643	648	666	699	734	751
Avg CPU Usage (%)	59.74	62.07	63.36	63.89	64.35	64.71
	25.64	28.21	28.60	28.20	28.22	28.72
Avg # of Threads Spawned	542	555	553	554	558	562
	598	606	617	624	627	632
UI Responsiveness (%)	98	98.41	98.22	97.95	97.83	97.80
	75.33	71.25	67.94	64.70	62.43	61.61
UI Freeze (%)	2	1.58	1.77	2.04	2.81	2.19
	24.66	28.75	32.05	35.29	37.56	38.38
UI Lag (%)	0.33	0.16	0.11	0.16	0.2	0.16
	2.5	3.08	2.94	2.62	2.53	2.36

The experimental results that are presented in Table 8.5 clearly display the huge performance improvement when we set garbage collector mode to *server*. The gray highlighted rows present the results obtained when garbage collector mode is set to *server*, and other rows show the results for the default case. As it can be seen from the table, this mode change boosts the number of crawled URLs by 336%, and improves the UI responsiveness by 58.74%.

8.1.6. Concurrent Connections Limit Experiment

In .NET framework the limit for the number of concurrent connections per host is two by default. However, crawlers are more likely to need more than two connections at the same time for each host when crawling gigantic websites. EcCrawler distributes crawling tasks for each host individually by using pre-assigned values which are 10 by default. However with having 10 crawling tasks per host, it is quite hard to reach maximum consistent two connections per host limitation because of the application processing delays. Therefore, for this experiment, one of the biggest E-commerce websites alone is used with 300 concurrent crawling agents.

Table 8.6. Results for the concurrent connection limit experiment

Concurrent Connections Limit (Minutes)	5	10	15	20	25	30
Crawl Success	36,804	76,486	118,736	145,322	175,070	203,082
	24,841	54,223	83,540	110,386	138,644	178,467
Crawl Fail	57	76	95	100	104	105
	0	3	4	4	4	4
Avg # of Pages per Minute Crawl	7,265	7,588	7,865	7,221	6,961	6,731
	4,898	5,375	5,532	5,488	5,517	5,919
Avg # of Processed Pages per Minute	968	823	761	708	695	676
	781	779	734	698	680	674
Avg Download Speed Per Second (KiloBytes)	1,582	1,643	1,699	1,558	1,501	1,450
	1,075	1,167	1,194	1,182	1,185	1,269
Avg Upload Speed Per Second (KiloBytes)	47	48	49	45	43	41
	30	32	33	33	33	35
Discovered Unique URLs	87,375	126,071	154,488	174,936	200,291	222,008
	70,057	110,788	136,557	156,776	176,680	203,628
Avg RAM Usage (MegaBytes)	763	798	829	862	884	903
	734	797	839	859	870	882
Avg CPU Usage (%)	54.84	54.18	54.16	47	44.01	41.26
	31.08	33.89	33.98	33.46	33.11	36.67
Avg # of Threads Spawned	568	567	558	565	567	570
	571	569	569	570	570	563
UI Responsiveness (%)	98	98.08	98	97.91	97.86	97.86
	98.83	98.66	98.72	98.79	98.8	98.8
UI Freeze (%)	2	1.91	2	2.08	2.13	2.13
	1.66	1.33	1.27	1.2	1.2	1.194
UI Lag (%)	0	0	0	0.08	0.06	0.08
	0	0.16	0.16	0.12	0.13	0.11

How to increase the maximum limit and the other parameters are shown in Appendix Code Snippet 9. The results of this experiment are given in Table 8.6, where gray highlighted rows show the results of the best performance settings (i.e., maximum consistent connections per host limit is 1000) and the other rows display results of the same settings except that the connection limit is 2. When the results given in Table 8.6 are analyzed, there is a significant performance boost in the first 5 minutes. Then, this performance gain decreases gradually, because, the number of the URLs to be discovered decreases as a greater portion of the website is crawled. At the 30th minute, since the majority of the site were already crawled, the number of the crawled pages almost become equal for the two methods because there were not left many pages to crawl.

8.1.7. All Default Settings Experiment

In the final experiment, the application is run without any advanced optimizations to show the effects of the proposed methods. To disable all of the optimizations, *HTTP Compression* and *KeepAlive* facilities are removed from the web page fetcher function; thread pool initialization and the limit on the maximum consistent connection per host are removed; and finally, garbage collector mode setup is removed from the *APP.config*. The results of this experiment are presented in Table 8.7 where the gray highlighted rows display the results of the best performance settings while the other rows show the results of all default settings.

When Table 8.7 is analyzed, a lot of significant improvements in the application performance can be observed. The proposed configurations in this study increase the number of crawled pages by 472%, decrease the UI freeze by 95.77%, reduce the UI lag by 96.39%, and improve the UI responsiveness by 102% with respect to default settings. Overall application performance is improved and the application is able to utilize the available system resources.

Table 8.7. Results for all default settings experiment

All Default Settings (Minutes)	5	10	15	20	25	30
Crawl Success	30,917	62,595	98,092	134,875	171,821	210,189
	4,232	8,628	14,546	21,628	29,445	36,728
Crawl Fail	114	459	1,348	1,622	1,697	1,754
	57	81	105	151	233	325
Avg # of Pages per Minute Crawl	6,086	6,195	6,485	6,693	6,827	6,960
	800	818	934	1,038	1,127	1,173
Avg # of Processed Pages per Minute	815	805	780	738	715	704
	295	395	463	514	547	555
Avg Download Speed Per Second (KiloBytes)	2,536	2,550	2,665	2,633	2,631	2,660
	1,146	1,117	1,233	1,348	1,437	1,458
Avg Upload Speed Per Second (KiloBytes)	38	39	42	44	44	45
	3	3	4	4	5	5
Discovered Unique URLs	153,994	207,914	259,737	309,515	346,910	383,869
	45,317	62,624	79,247	90,788	103,691	110,341
Avg RAM Usage (MegaBytes)	1,003	1,106	1,177	1,229	1,275	1,292
	787	942	918	880	856	869
Avg CPU Usage (%)	59.74	62.07	63.36	63.89	64.35	64.71
	17.80	19.74	21.64	23.15	24.29	24.66
Avg # of Threads Spawned	542	555	553	554	558	562
	369	450	470	483	488	497
UI Responsiveness (%)	98	98.41	98.22	97.95	97.83	97.80
	18.16	20.08	32.22	40.66	45.43	48.19
UI Freeze (%)	2	1.58	1.77	2.04	2.81	2.19
	81.83	79.91	66.77	59.33	54.56	51.80
UI Lag (%)	0.33	0.16	0.11	0.16	0.2	0.16
	9.5	8.91	6.66	5.54	5.03	4.44

8.2. Sentiment Analysis – Classification Experiments

8.2.1. Experimental Setup

System development tests are done only on the dataset which is collected by us in this study. The publicly published datasets of the previous literature studies are tested with only certain promising configurations due to lack of resources. Thus, they may perform better with further parameter optimization. We provide these datasets as well on the Github repository and present their results in Section 8.2.3.

We have experimented with the following SVM parameters:

- **Shrinking:** True or False
- **SVM Type:** C-SVC or nu-SVC
- **Kernel Type:** Linear, Polynomial, Radial Basis Function, Sigmoid
- **Kernel Parameters:** **d:** Degree, **g:** gamma, **r:** coef0, **c:** cost, **n:** nu, **p:** epsilon

More information about these parameters can be obtained from the official document⁵⁰ of LibSVM framework. In addition to these parameters, we have tested L1, L2, L3, L4 and L5 vector normalizations⁵¹ of *SVMNodes* provided by LibSVMSharp wrapper.

For document vector computation, we have tested the following metrics:

- **Weighting Scheme:** TF, IDF, TF*IDF, RF, TF*RF
- **TF Variants:** Binary, Raw Frequency, Log Normalization, Double Normalization, and BM25

⁵⁰ <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>

⁵¹ [https://en.wikipedia.org/wiki/Norm_\(mathematics\)](https://en.wikipedia.org/wiki/Norm_(mathematics))

- **IDF Variants:** Unary, IDF, IDF Smooth, IDF Max, Probabilistic IDF, Delta IDF, BM25 IDF, and Delta BM25 IDF
- **BOW methods:** 1-Word Gram, N-character Grams, 1-Word Gram + N-character Grams

When implementing Delta IDF to our system, we had to make a modification. Proposed Delta IDF method causes error when N_{nt} or N_{pt} is zero. Therefore, if $N_{nt} = 0$, we set the IDF as $\log(N_{pt})$ and if $N_{pt} = 0$, we set IDF as $\log(N_{nt})$. For BM25 we have chosen $k_1=1.2$ and $b=0.95$ which are default values according to (Jones, et al., 2000).

For TF weighting scheme; all TF formulations are tested; for TF*IDF weighting scheme all combinations of TF and IDF formulations are experimented; for RF scheme, RF is implemented as in equation 14 and all combinations of TF*RF are tested. All of the algorithms are implemented in C#.

Feature selection techniques are also extensively tested on our dataset. In vectoral representation of text documents, each feature is a dimension thus, feature selection is also known as dimension reduction. For selecting features, top N scored features by a metric such as IG and CHI2 are selected, and the rest of the features are discarded. Feature selection increases the classification performance by reducing the complexity of the problem and can improve the success rate by eliminating noise features. According to our experimental analysis, there is a tradeoff between the performance and the success rate when feature selection is applied, and there isn't a clear pattern for success improvement. For feature selection, we have employed standard definitions of IG and CHI2 algorithms. We refer to the study of Lan and colleagues (2009) for more information about IG and CHI2 algorithms.

8.2.2. Experimental Results and Discussions

We have experimented with every combination of shrinking, SVM types, kernel types, and the parameters of kernels. According to our empirical analysis, shrinking does not affect the results significantly and selecting the default mode as “true” performs the best. For SVM type; C-SVC, for normalization; L2 normalization perform the best. Among the kernel types, the best performing kernel type is Linear kernel. It is fast when compared to the other kernels, and it produces better success rates. The possible explanation of this is since text categorization task already contains so many dimensions, mapping them into higher dimension by using the other kernels decreases the success rate and increases the complexity of the problem. According to our experiments, parameter tuning of the other kernels is not practical due to the massive number of features when thousands of text documents are being classified. Thus, obtaining higher success rates via parameter tuning is harder in the other kernels when compared to the Linear kernel in sentiment analysis field.

After the first experiments have been completed, we have decided to continue experimenting with only Linear kernel, C-SVC as SVM type and none or L2 normalization. At the second stage of the experiments, we have tested every combination of the TF*IDF weighting metrics that are presented in Section 8.2.1. We present the best performer configuration for each one of the TF and IDF metrics according to accuracy and F-measure values in Table 8.8 where Norm is Normalization, Acc is Accuracy and F is F-measure. The first row in the table shows the best configuration which uses TF*IDF weighting scheme, where TF is implemented as Log Normalization and IDF is computed by Probabilistic IDF formula, with L2 normalization. The worst performer weighting scheme, which is given in the last row of the table, uses raw frequency as TF, and Delta BM25 IDF as IDF without any normalization.

Table 8.8. Best product comments classification results of each TF and IDF weighting metrics on our collected dataset

TF	IDF	Norm	Acc	F
Log Normalization	Probabilistic IDF	L2	91.08	91.22
Log Normalization	BM25	L2	90.91	91.02
BM25	Probabilistic IDF	L2	90.83	90.97
BM25	IDF Smooth	L2	90.75	90.83
Log Normalization	IDF	L2	90.75	90.82
Double Normalization	Probabilistic IDF	L2	90.75	90.89
Raw Frequency	IDF Smooth	L2	90.58	90.64
Binary	IDF Smooth	L2	89.91	90.03
Binary	Delta IDF	L2	89.75	89.82
BM25	IDF Smooth	None	89.50	89.50
BM25	RF	None	88.25	88.78
Log Normalization	IDF Max	L2	87.66	87.61
Raw Frequency	RF	None	86.83	87.16
BM25	Delta BM25 IDF	None	86.5	86.23
Raw Frequency	Delta BM25 IDF	None	84.66	83.86

Next, we have experimented N-gram transformations in two different settings with a combination of all of the weighting metrics presented in Section 8.2.1. In the first mode, the documents are transformed into N-gram literals and in the second mode, the documents are transformed into unigram word + N-gram (2, 3, 4) literals representation. At the end of the experiments, we observed that none of the experimented N-gram configurations are more successful than the best configuration presented in Table 8.8 which do not involve any N-grams. Thus, we propose that in Turkish sentiment analysis of e-commerce product reviews, using N-grams does not lead to any improvement. The results of N-gram experiments are presented in Table 8.9.

Table 8.9. N-gram preprocessing effect on Turkish E-commerce product comments sentiment analysis

Configuration	Accuracy	F-measure
Best configuration	91.08	91.22
Character 2-gram	74.50	74.37
Character 2-gram + Unigram words	86.75	86.83
Character 3-gram	85.50	85.71
Character 3-gram + Unigram words	88.83	89.05
Character 4-gram	89.16	89.33
Character 4-gram + Unigram words	89.91	90.06

Furthermore, when N is chosen as 4 or bigger, the dimension of the problem notably increases since the number of features is increased as shown in Table 8.10. Thus, the complexity and the processing times significantly increase. Additionally, when a lower N value is chosen, the number of distinctive features between documents decreases, therefore, SVM classifier takes much more time to converge when training the model.

Table 8.10. Number of features in the entire product comments dataset when N-gram method is applied

Unigram words	Character N-grams					
	2-gram	3-gram	4-gram	5-gram	6-gram	7-gram
11,058	1,165	8,532	32,612	75,422	119,284	154,698

We have made extensive spelling correction tests as well. To employ efficient spelling correction, a very fast dictionary system which uses Symmetric Delete spelling correction algorithm is implemented. The dictionary framework is publicly published by Wolf Garbe⁵². The framework provides top N suggestions

⁵² <https://github.com/wolfgarbe/sympell>

based on maximum edit distance parameter. Our tests involve all combinations of the following options:

- *Dictionary datasets*⁵³: Hunspell, Dict_Aff
- *Dictionary with suffix*: True, False
- *Maximum edit distance*: 0, 1, 2, 3, 4
- *Add top N suggestions to the document*: 0, 1, 2, 3, 4, 5
- *Discard word if not exists in the dictionary*: True, False

However, among the 240 ($2 \times 2 \times 5 \times 6 \times 2 = 240$) different configurations shown above, none of the configurations have improved the accuracy of the best configuration. Thus, we propose that in Turkish product comments sentiment analysis, spelling correction does not improve the accuracy even fine tuning of the parameters are made.

Our next experiments are about testing the effects of feature selection. We have experimented all combination of the following settings with the best configuration which is shown in the first row of Table 8.8. IG and CHI2 methods are used to select top N percent features where

- N is between 0.1%-10% with 0.1% increment
- N is between 10%-99% with 1% increment

The results of the feature selection experiments are presented in Figure 8.2, Figure 8.3, Figure 8.4 and Figure 8.5 where in Figure 8.2 and Figure 8.3, the X axis is the top N% feature selection ratio and the Y axis is the obtained accuracy, and in Figure 8.4 and Figure 8.5 the X axis is top N% feature selection ratio and the Y axis is the ratio of documents which become empty after feature selection.

⁵³ <https://code.google.com/archive/p/tr-spell/>

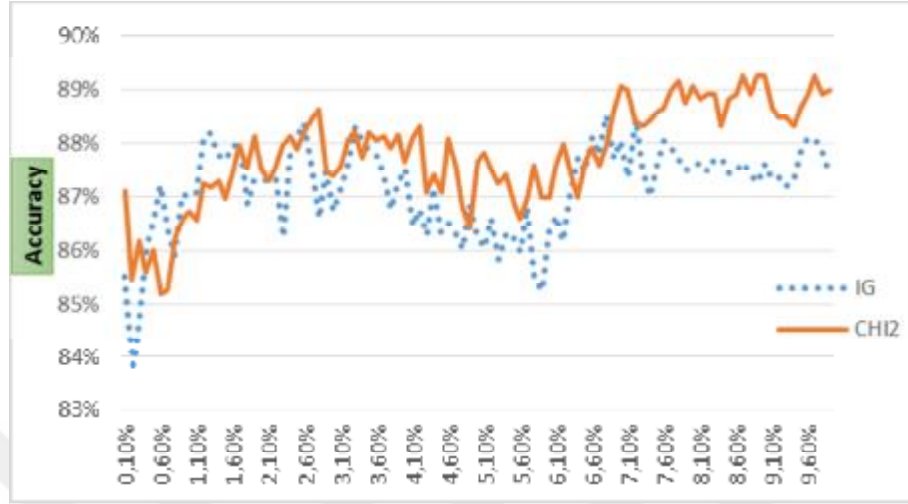


Figure 8.2. Classification accuracy when top N% (0.1% - 10%) features are selected

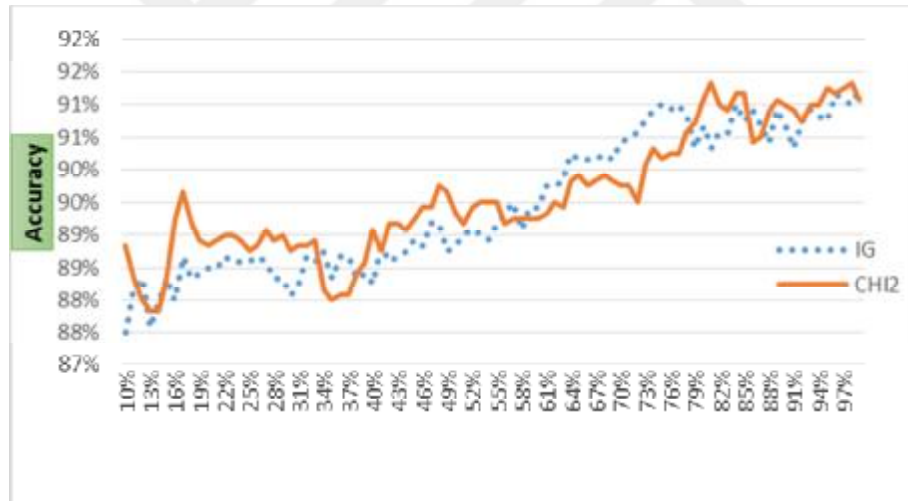


Figure 8.3. Classification accuracy when top N% (10% - 97%) features are selected

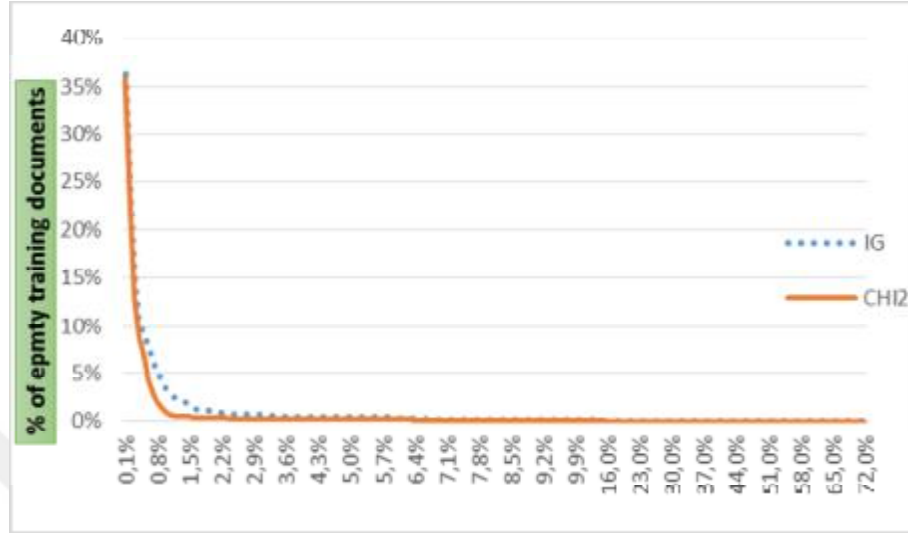


Figure 8.4. Percentage of training documents that become empty when top N% feature selection is applied

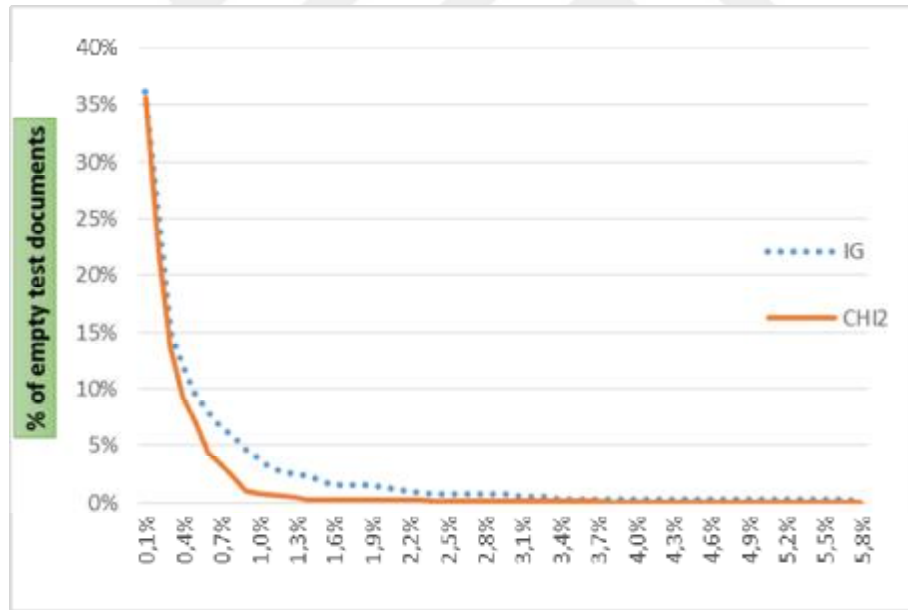


Figure 8.5. Percentage of test documents that become empty when top N% feature selection is applied

When the best configuration in Table 8.8 is used without any feature selection, our dataset is classified with 91.08% accuracy. When top 0.1%-10% features are selected with 0.1% increment by using the IG and CHI2 feature selection methods, the best-obtained accuracy of IG is 88.50% where N is 2.6%, and the best-obtained accuracy of CHI2 is 89.25% where N is 8.7% as shown in Figure 8.2. Figure 8.3 displays classification accuracy of IG or CHI2 based feature selection when top 10%-99% features are selected with 1% increment. The best-obtained accuracy of IG is 91.25% where N is 99%, and best-obtained accuracy of CHI2 is 91.33% where N is 81%. As shown in Figure 8.2 and Figure 8.3, if we select very small number of features, classification accuracy decreases, however if enough sized feature subsets are chosen classification accuracy increase with respect to no feature selection case. Performances of IG and CHI2 feature selection methods are very similar; however CHI2 has slightly better feature selection performance.

Figure 8.4 and Figure 8.5 display percentage of training and test documents which do not have any features when feature selection is applied. The values are average of 10-fold cross-validation tests. When IG based feature selection is made, at the very least 72% of the top features need to be selected to utilize all of the documents in the training set, and when CHI2 based feature selection is made, 73% of the features need to be selected. According to Figure 8.5 when IG based feature selection is made, at the very least 5.8% of the top features need to be selected to classify all of the documents in the test set, and when CHI2 based feature selection is made, 3.8% of the features need to be selected. Number of predictive features or training features that are left to be used when either training the model or classifying the test data, is an important issue when applying feature selection. However, according to our observations, in the literature, this issue is highly ignored and not mentioned. We predict that researchers ignore training documents that become empty after feature selection is made. We believe that the classifier software that is used masks this problem. For example, WEKA which is commonly

used in literature by the researchers for classification experiments, assigns empty test documents to the class having the highest number of instances. Therefore, researchers probably not even being aware of this problem and moreover, this default behavior can be very misleading when the class sizes are not balanced.

8.2.3. Comparison of Previous Works with this Study

Finally, we have tested our most successful weighting configurations with the ones which are found as successful in the literature. Therefore we have compared five configurations that are given in Table 8.11 over the publicly available datasets which are English product reviews (Blitzer, et al., 2007) (Books, DVD, Electronics, Kitchen, All_Merged), Turkish movie reviews (Demirtas & Pechenizkiy, 2013), Turkish product reviews (Demirtas & Pechenizkiy, 2013) (Books, DVD, Electronics, Kitchen, All_Merged), English movie reviews (Pang & Lee, 2004), and sentences of English movie reviews (Pang & Lee, 2005). In Table 8.11, Config_1 and Config_5 are our proposed most successful document vector computation methods; Config_2 and Config_4 are the best performer methods that are proposed by Paltoglou and Thelwall (2010), and Config_3 is the method that is found as successful by Lan and colleagues (2009).

Table 8.11. List of the tested product comments classification configurations on the other datasets

	TF	IDF	Norm	Feature Selection
Config_1:	Log Normalization	Probabilistic IDF	L2	None
Config_2 (Paltoglou & Thelwall, 2010):	BM25	Delta BM25 IDF	None	None
Config_3 (Lan, et al., 2009):	BM25	RF	None	None
Config_4 (Paltoglou & Thelwall, 2010):	BM25	Delta IDF	None	None
Config_5:	Log Normalization	Probabilistic IDF	L2	CHI2 with Top 1% Features

Table 8.12. Sentiment analysis experimental results for English and Turkish movie reviews datasets

	English Movie Reviews (Pang & Lee, 2004)		English Movie Reviews (Pang & Lee, 2005)		Turkish Movie Reviews (Demirtas & Pechenizkiy, 2013)	
	Accuracy	F	Accuracy	F	Accuracy	F
Config_1	86.70	86.67	74.45	74.14	88.21	88.15
Config_2	81.05	81.28	73.69	73.03	85.72	85.27
Config_3	84.55	84.98	73.47	74.12	86.13	86.43
Config_4	85.85	85.97	75.68	75.75	87.46	87.40
Config_5	82.35	82.52	69.02	69.02	87.41	87.52

Table 8.12 presents results when the five methods are applied to English and Turkish movie review datasets that are proposed by (Pang and Lee, 2004; Pang and Lee, 2005) and Demirtas and Pechenizkiy (2013) where F is F-measure. The

best accuracy and F-measure values for each dataset are written in boldface. As shown in the table, for two out of three movie review datasets, our proposed Config_1 method performs the best. Only for one English movie review dataset (Pang & Lee, 2005), Config_4 has the best performance; however, our proposed Config_1 is the second best and has very close performance to Config_4.

Table 8.13. Experimental results for English product reviews dataset (Blitzer, et al., 2007)

	Books		DVD		Electronics		Kitchen		All Categories	
	Acc.	F-M.	Acc.	F-M.	Acc.	F-M.	Acc.	F-M.	Acc.	F-M.
Config_1	84.95	85.07	86.45	86.23	87.95	87.83	90.85	90.83	88.33	88.30
Config_2	81.25	79.29	81.10	78.82	86.25	85.37	88.95	88.67	84.12	82.89
Config_3	80.50	81.72	81.05	81.96	85.20	85.63	87.45	87.94	84.82	85.29
Config_4	83.20	83.16	84.70	84.52	86.45	86.17	89.20	89.24	87.13	87.04
Config_5	80.15	82.25	81.35	81.28	84.64	84.63	85.79	85.85	83.13	83.17

Table 8.14. Experimental results for Turkish product reviews dataset (Demirtas & Pechenizkiy, 2013)

	Books		DVD		Electronics		Kitchen		All Categories	
	Acc.	F-M.	Acc.	F-M.	Acc.	F-M.	Acc.	F-M.	Acc.	F-M.
Config_1	81.92	81.85	75.35	75.73	80.92	80.50	77.07	76.11	83.58	83.46
Config_2	78.50	76.66	67.57	63.21	77.28	77.23	73.64	73.60	81.15	80.45
Config_3	79.50	80.16	74.28	75.15	76.71	77.57	73.35	74.24	82.11	82.40
Config_4	77.07	76.57	72.64	71.99	77.14	76.96	76.14	75.74	82.85	82.57
Config_5	80.62	80.63	75.05	74.98	77.59	75.77	75.61	74.07	83.97	83.91

In Table 8.13 and Table 8.14, experimental results for the English product reviews (Blitzer, et al., 2007) and Turkish product reviews (Demirtas & Pechenizkiy, 2013) datasets are presented. When the results in Table 8.13 and Table 8.14 are analyzed, it is easily observed that our proposed method which is Config_1 performs the best for all categories. Only for Turkish product reviews dataset, when all categories that are books, DVD, electronics, and kitchen are

merged, our second proposed method which is Config_5 has the best performance, and as it is seen from Table 8.14, the difference between Config_1 and Config_5 in terms of accuracy and F-measure is very small and Config_1 is the second best for this case as well. It is also notable that, when all categories are merged for both product review datasets, all five methods perform slightly better. This performance improvement may be due to the fact that when we merge all categories, the number of training instances increases and this allows us to learn better classification model.

In our experiments, we observed that there is a significant difference between our obtained results and the results computed by Paltoglou and Thelwall (2010) for Delta IDF (i.e., Config_4) and Delta BM25 IDF (i.e., Config_2) methods even on the same dataset. For example, Config_2 obtains 96.90% accuracy on the English Movie Reviews (Pang & Lee, 2004) dataset and 96.40% accuracy on the English Product Reviews (Blitzer, et al., 2007) dataset in the study of Paltoglou and Thelwall (2010), however we obtain 81.05% and 81.25% accuracy respectively with the same configuration for the same datasets. Additionally, RF weighting scheme (i.e., Config_3), which is shown as superior to the traditional TF*IDF method by Lan and colleagues (2009), has never surpassed the success rate of the best TF*IDF configuration in any of the conducted experiments. We believe that the reason for the failure of these three mentioned metrics can be the differences in how the feature extraction and feature selection is made, which data preprocessing method is applied, or how no-feature left documents are handled. These differences in the results also show the importance of making formatted datasets publicly available as we do in this study because when future research is done over them, all researchers will use exactly the same settings. Thus, the results will be much more objectively comparable.

8.3. Record Linkage - Clustering Experiments

8.3.1. Experimental Setup

We have conducted many different parameter/metric/algorithm experiments due to the reason that in record-linkage, there is not a single algorithm or methodology that performs best on all domains. Additionally, how the features are tokenized is extremely important in distance calculation of strings. The tokenization process is assigning best representative numerical weights to the each feature in the vector form. Therefore, we have tested the following weighting metrics to find out which one may perform best on products domain:

- **Vector Normalization:** None, L1, L2
- **Weighting Scheme (WS):** TF, IDF, TF*IDF
- **TF Variances:** Binary, Raw Frequency, Log Normalization, Double Normalization, BM25
- **IDF Variances:** Unary, IDF, IDF Smooth, IDF Max, Probabilistic IDF
- **BOW:** 1-Word Gram, N-character Grams, 1-Word Gram + N-character Grams

An extensive study of the above metrics is presented by Paltoglou and Thelwall (2010) where we have taken the formulations of the BM25 TF schemes. Recall that variants of TF is presented in Section 3.4.1 and variants of IDF are presented in Section 3.4.2.

For TF WS; all TF variants are tested, for TF*IDF; all combinations of TF and IDF variants are tested. We have not employed any feature selection since features are already sparse and valuable unlike in other text classification or clustering domains. As a distance metric to calculate the distance between record pairs, we have tested the algorithms presented in Table 3.4. As shown in Table 3.4,

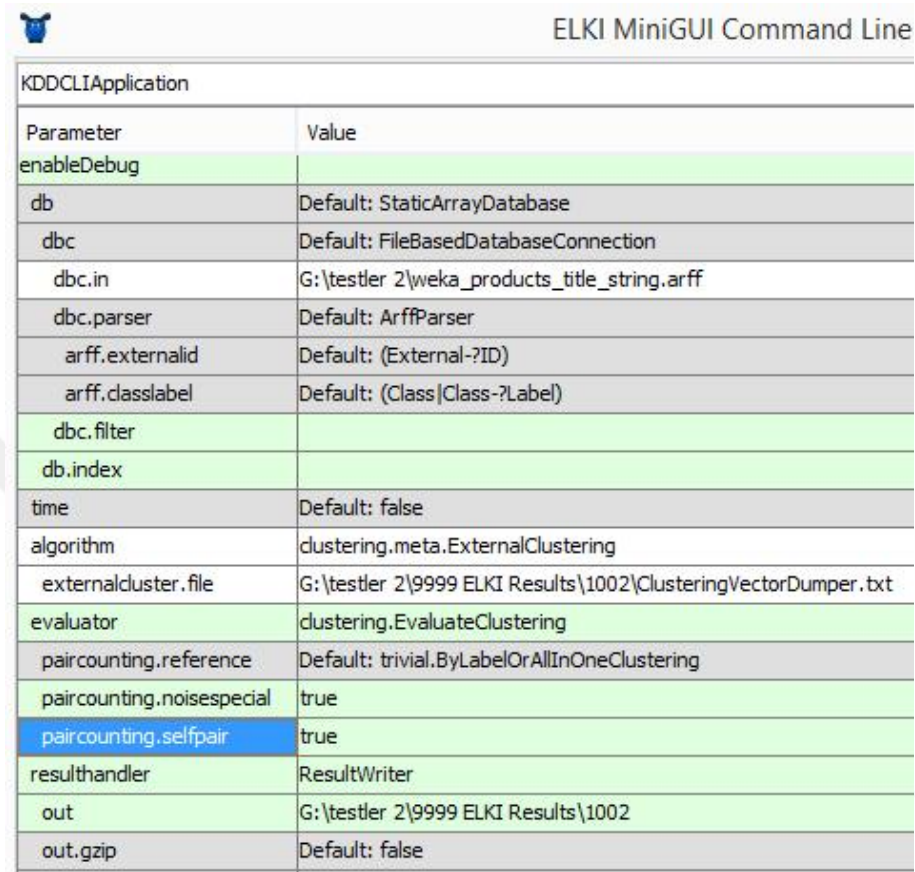
the range of the each distance algorithm changes. Additionally, some of them do not have a finite upper limit. Because of this reason, for each algorithm, the individual range of distance values needs to be chosen when experimenting.

Since each cluster can have multiple records, we have experimented with multiple different linkage algorithms that are shown in Table 3.5.

Another very controversial area of clustering is, how the results will be evaluated. There does not exist any standard or widely accepted metric to measure the performance of clustering process. Additionally, even the implementation of the same metric may differ in different studies. For example for the F-measure metric, we have found at least five different algorithms that are used. Since there is no single implementation of the same metric in the previous studies, the presented results in the previous works are not precise if the used implementation is not clearly mentioned. According to our analysis, in many of the previous works, the implementation of the used metrics are not presented.

When developing our system, we have used the F-measure algorithm presented by Manning and colleagues (2009) to evaluate our results. However, recall that all of the scores we present in this study for evaluating clustering success rate are calculated by using ELKI. The settings of score calculation in ELKI also does matter. How we calculate the scores is presented in Figure 8.6. Additionally, the command argument that is used to calculate evaluation is shown below:

```
KDDCLIApplication -dbc.in "input.arff" -algorithm  
clustering.meta.ExternalClustering -externalcluster.file  
"inputClusteringVectorDumper.txt" -evaluator clustering.EvaluateClustering -  
paircounting.noisespecial -paircounting.selfpair -resulthandler ResultWriter -out  
"outputDirectory"
```



The image shows a screenshot of the 'ELKI MiniGUI Command Line' window. It contains a table with two columns: 'Parameter' and 'Value'. The table lists various settings for the ELKI application, including database settings, file paths, evaluation methods, and output options. The 'paircounting.selfpair' parameter is highlighted in blue.

Parameter	Value
enableDebug	
db	Default: StaticArrayDatabase
dbc	Default: FileBasedDatabaseConnection
dbc.in	G:\testler 2\weka_products_title_string.arff
dbc.parser	Default: ArffParser
arff.externalid	Default: (External-?ID)
arff.classlabel	Default: (Class Class-?Label)
dbc.filter	
db.index	
time	Default: false
algorithm	clustering.meta.ExternalClustering
externalcluster.file	G:\testler 2\9999 ELKI Results\1002\ClusteringVectorDumper.txt
evaluator	clustering.EvaluateClustering
paircounting.reference	Default: trivial.ByLabelOrAllInOneClustering
paircounting.noisespecial	true
paircounting.selfpair	true
resulthandler	ResultWriter
out	G:\testler 2\9999 ELKI Results\1002
out.gzip	Default: false

Figure 8.6. An example of the settings that are used to calculate clustering evaluation by ELKI

When the above settings are used, F-measure under the *Pair counting measures* section of the results which is calculated by ELKI produces the same result as the F-measure algorithm presented by Manning and colleagues (2009). We have found that *Pair counting measures* is the most fitting approach for calculating the success of the clustering in record linkage area after our empirical analysis. Therefore, all of the F-measures presented in this thesis are calculated by using ELKI's evaluation and the *Pair counting measures* approach. An example of evaluation results generated by ELKI displayed in Figure 8.7.

```

1 # Pair counting measures
2 Jaccard 0.2597482979162368
3 F1-Measure 0.41238126433016703
4 Precision 0.9862906384645516
5 Recall 0.26068951237188115
6 Rand 0.9951515479102199
7 ARI 0.4107734749769209
8 FowlkesMallows 0.5070657014611374
9 # Entropy based measures
10 NMI Joint 0.8102981093226196
11 NMI Sqrt 0.9000344584388756
12 # BCubed-based measures
13 F1-Measure 0.5836728501804221
14 Recall 0.9943126793795009
15 Precision 0.41307682074686025
16 # Set-Matching-based measures
17 F1-Measure 0.6774862144326119
18 Purity 0.5136548518303323
19 Inverse Purity 0.9947704822777517
20 # Editing-distance measures
21 F1-Measure 0.46126330065998566
22 Precision 0.38524113887274836
23 Recall 0.5746658919233004
24 # Gini measures
25 Mean +-0.3443 0.7036947500631808

```

Figure 8.7. An example clustering evaluation results generated by ELKI

8.3.2. Experimental Results and Discussions

Since there are thousands of different combinations of the available algorithms along with the hundreds of different parameter options and we have had limited time and resources, we have used an iterative empirical approach for conducting the tests. Additionally, we have used a lot of time and resources on the new algorithm development that has been all failed to improve the success rate such as spell correction reinforcement, blocking reinforcement via an SVM-based

classification, success improvement via additional similarity based thresholding. All of the experiments are conducted on the DSE dataset.

The iterative empirical approach we propose is based on splitting the possible configurations into smaller logical stages, conducting experiments with these stages and then conducting the final experiments with the promising configurations that are obtained in the previous stages. This approach still does not ensure the best possible outcome. However, it is one of the most effective approaches when it is not possible to experiment with all of the possible configurations and the parameters.

In the first stage, we have experimented to find which combination of vector normalization, BOW transformation, the maximum price ratio of price matching system and the weighting scheme are promising. So all combination of the vector transforms, TF or TF*IDF weighting schemes, TF variances, IDF variances and the BOW approaches that are presented in Section 8.3.1 are tested. However, these tests still require distance metric, distance threshold and the linkage criteria to be chosen. Also, our proposed matching system configurations are necessary. So it is not possible to find which combination performs best unless all combinations are tested. For these untested configurations and parameters, we have used the ones that we have found most useful during the initial system development. After our through experiments are done in the first stage, we have chosen a single most promising configuration to carry experimenting on the other stages due to lack of time and resources we have had. The most promising configuration we have found during the first stage experimenting is shown in Table 8.15.

Table 8.15. Most promising configuration of the first stage experiments

Method Name	Methodology
Linkage Criteria	UPGMA (Average Linkage Clustering)
Distance Metric	Squared Euclidean Distance
Vector Normalization	L2
Weighting Scheme	TF*IDF
IDF Variant	IDF Smooth
TF Variant	BM25_TF
Max Distance to be Paired Threshold	1.45
Maximum Price Ratio	1.5
BOW Method	Word based 1-gram + character based 2-gram

In the second stage experimenting, we have tested every configuration of distance algorithms and the linkage criteria that are shown in Table 3.4 and Table 3.5, respectively. Since the range of the each distance algorithm changes, the algorithms works only with the proper distance parameter. Therefore, we have conducted the experiments with a different range of distance thresholds the each distance metric. For the ones that have infinite range, we have checked the farthest distance having records in our dataset and based on that distance value we have set the upper limit. Then we have split this range between 0 and the determined upper limit to 100 equal intervals, and then each interval is tested. For the upper limit defined ranges such as Dice Distance, which is between 0-1 always, we have applied the same approach. For example for the Euclidean Distance metric, the farthest distance value between the records in our dataset is 1.4. Thus, we have decided upper limit as 2.0 and tested all values between 0 and 2 with 100 equal intervals. In this case the tested values are 0, 0.02, 0.04, 0.06 ... 2.0. This approach ensured that each algorithm was tested as much as equally and provided more objective results. The results of the top performing configuration for each distance

metric is shown in Table 8.16 where DT is distance threshold, F is F-measure, P is precision, and R is Recall.

Table 8.16. Experimental results of distance metric, linkage criteria and distance threshold for record linkage task

Configuration <i>Dataset: DSE</i>			F	ARI	P	R
Distance Metric	Linkage Criteria	DT				
Average Kullback Leibler Divergence Distance	Complete	1.88	38.98	38.46	29.99	55.64
Bray Curtis Distance	Average	0.69	95.97	95.94	98.17	93.87
Canberra Distance	Average	54.0	79.19	79.05	77.45	81.00
Cosine Distance	Centroid	0.61	96.45	96.42	98.10	94.85
Dice Distance	Average	0.60	95.05	95.02	96.86	93.31
Euclidean Distance	Average	1.20	96.09	96.06	96.62	95.55
Tanimoto Non-Binary Distance	Centroid	0.77	96.22	96.20	97.70	94.79
Hamming Distance	Wards	2.0	54.57	54.16	40.50	83.60
Jaccard Distance	Average	0.75	94.88	94.84	96.77	93.06
Manhattan Distance	Wards	0.80	92.68	92.63	96.90	88.80
Maximum Distance	Wards	0.79	92.84	92.79	97.31	88.76
Pearson Correlation Distance	Centroid	1.04	96.13	96.10	99.07	93.35
Squared Euclidean Distance	Average	1.44	96.23	96.21	96.78	95.69
Normalized Canberra Distance	Average	0.77	93.88	93.84	97.19	90.78

In the third stage, we have experimented with the *minLenght*, *editDistanceProductCode*, *minSubstringRatio*, *minLCSRatioWordLenght* and *minLongestCommonRatio* parameters of the product code detection algorithm that is shown in Figure 6.3. This time, we have used the best configuration that we have obtained in Table 8.16. We have obtained best 97.12% F-measure, 97.10% ARI, 99.57% Precision, and 94.79% Recall. The details of the product code experiments are shown in Table 8.17.

Table 8.17. Results of parameter experiments for the product code detection system

Parameter	Tested Values	Best Performing Value
<i>minLenght</i>	3, 4, 5, 6, 7	4
<i>editDistanceProductCode</i>	0, 1, 2, 3, 4, 5	2
<i>minSubstringRatio</i>	20%, 30%, 40%, 50%, 60%	20%
<i>minLCSRatioWordLenght</i>	3, 4, 5, 6, 7, 8	8
<i>minLongestCommonRatio</i>	40%, 50%, 60%, 70%, 80%	40%

Then we have experimented with the most promising configurations that we have obtained in the first three stages of the experiments. We have selected the Jaccard Distance, Pearson Correlation Distance, Cosine Distance, Bray Curtis Distance, Normalized Canberra Distance, Dice Distance, and Tanimoto Non-Binary Distance configurations since they all perform with over 85% precision, and their range is in finite ranges. Thus, they are easier to apply different domains or datasets. As a result of final experiments, we have obtained the best results that are shown in Table 8.18. In Table 8.18, WS is TF*IDF, TF variant is BM25, IDF variant is IDF Smooth, vector normalization is L2 for all of the experiments and *minLenght* is ML, *editDistanceProductCode* is DPC, *minSubstringRatio* is SR, *minLCSRatioWordLenght* is LRW, and *minLongestCommonRatio* is LCR.

Table 8.18. Results of the final stage parameter experiments for record linkage task

Configuration	F	ARI	P	R
DM: Jaccard Distance LC: Average Linkage Clustering, DT: 0.75, ML: 4, DPC: 2, SR: 10%, LRW: 8, LCR: 30%	95.67	95.64	97.98	93.47
DM: Pearson Correlation Distance LC: Centroid Linkage Clustering, DT: 1.04, ML: 4, DPC: 2, SR: 30%, LRW: 8, LCR: 30%	96.48	96.46	99.64	93.51
DM: Cosine Distance - Best LC: Centroid Linkage Clustering, DT: 0.61, ML: 4, DPC: 2, SR: 20%, LRW: 8, LCR: 30%	97.12	97.10	99.57	94.79
DM: Bray Curtis Distance LC: Average Linkage Clustering, DT: 0.69, ML: 4, DPC: 2, SR: 10%, LRW: 8, LCR: 30%	96.52	96.50	98.87	94.28
DM: Normalized Canberra Distance LC: Average Linkage Clustering, DT: 0.77, ML: 4, DPC: 2, SR: 10%, LRW: 8, LCR: 30%	94.88	94.84	98.60	91.42
DM: Dice Distance LC: Average Linkage Clustering, DT: 0.60, ML: 4, DPC: 2, SR: 10%, LRW: 8, LCR: 30%	95.67	95.64	97.98	93.47
DM: Tanimoto Non-Binary Distance LC: Centroid Linkage Clustering, DT: 0.77, ML: 4, DPC: 2, SR: 10%, LRW: 8, LCR: 30%	96.92	96.90	99.19	94.76

As a next step, we have experimented on our macro-average dataset with the best performing configuration. Additionally, we have tested many state-of-the-art clustering algorithms by using latest ELKI 0.7.1 and latest WEKA (Hall, et al., 2009) 3.7 frameworks on the macro-average dataset. The vectors for these tests are calculated by the best weighting configuration obtained in our own dataset

experiments with one critical difference. When the vectors are computed, the entire dataset is provided to the vectorization system since none of the experimented state-of-the-art algorithms work in an incremental fashion. Moreover, WEKA experiments never completed even though we had waited for 53 days. We have reported this issue to the developers since same algorithms on ELKI completes in less than 5 minutes, however, they have responded us that that is expected. An example of the command we have run on the WEKA is below with additionally Store clusters for visualization and Classes to clusters evaluation options are selected: `weka.clusterers.HierarchicalClusterer -N 250 -L AVERAGE -P -A "weka.core.EuclideanDistance -R first-last"`. Moreover, ELKI version 0.7.1 has a major glitch that breaks the ARFF⁵⁴ parser. We have discovered this issue while experimenting due to unusual low success rates of the algorithms. After we have reported this issue to the developer team of ELKI, they have fixed it on Github repository. After this fix, we have rebuilt the Github repository master branch with the following command: `"mvn -DskipTests -Dmaven.javadoc.skip=true -P svg,bundle package"`. We have to note that we have failed to compile on Windows 8.1 OS. Thus, we have used a fresh installed Ubuntu. After the compile, we have obtained "elki-bundle-0.7.2-SNAPSHOT.jar" bundle file. The date of the compiled source code is 6 June 2016. We also have included this bundle file to our dataset repository on the Github for the researchers who want to use the same bundle. The results we have obtained on the macro-average dataset is presented in Table 8.19. In Table 8.19, *Minclusters* and *K* are the number of clusters in the clustered dataset, *TIF* is TF*IDF weighting scheme which is shown in Table 8.15, and *Raw* is the raw term frequency where vector normalization is none and BOW methodology is only word based 1-gram. The number of clusters provided are the exact number of clusters since we have previously manually labelled each record and their TF*IDF weights are calculated based on the entire dataset. Both configurations generated

⁵⁴ Attribute-Relation File Format (ARFF), <http://weka.wikispaces.com/ARFF>

dataset files are available in the Github repository. Furthermore in Table 8.19, in each result cell, the first value is the minimum, the second value is average, and the third value is the maximum value of the obtained results in the tests that are conducted with the respected configuration. For example for K-Means Sort algorithm, 12.85 is minimum, 20.23 is average, and 35.15 is the maximum F-measure. We only present non-default parameters of the algorithms in the configuration except for AGNES as known as also HAC. In AGNES we have extra presented the linkage criteria and the distance metric as additional information. The results of clustering experiments conducted on the macro-average dataset is presented in Table 8.19 where RT is the average algorithm run-time for each test in seconds.

Table 8.19. Experimental results of the macro-average dataset clustering experiments

Configuration	F-measure		ARI		Precision		Recall		RT
	Raw	TIF	Raw	TIF	Raw	TIF	Raw	TIF	
Our System: DM: Cosine Distance LC: Centroid Linkage Clustering, DT: 0.61, ML: 4, DPC: 2, SR: 20%, LRW: 8, LCR: 30%	91.68 93.87 95.93	95.07 96.25 97.54	91.62 93.83 95.9	95.03 96.23 97.52	91.06 95.29 97.6	97.03 99.23 99.87	90.09 92.51 95.01	91.27 93.47 95.84	49.6
Affinity Propagation (Frey & Dueck, 2007)	32.06 41.38 52.39	76.68 79.95 84.52	31.39 40.84 52	76.53 79.82 84.42	20.53 29.83 41.72	81.84 86 89.53	63.02 68.62 78.3	68.83 74.75 82.56	27.7
Extract Flat Clustering From Hierarchy + AGNES (Kaufman & Rousseeuw, 2008) + Single Linkage + Squared Euclidean Distance, Minclusters	2.95 4.97 7.36	61.6 72.48 81.42	1.7 3.75 6.16	61.25 72.26 81.27	1.51 2.57 3.87	45.78 59.55 71.76	70.24 74.07 80.47	90.35 93.13 94.93	22.5
Extract Flat Clustering From Hierarchy + AGNES + Complete Linkage + Squared Euclidean Distance, Minclusters	35.56 53.8 67.48	84.59 87.87 91.95	34.93 53.43 67.25	84.49 87.79 91.9	23.9 44.03 63.07	82.76 87.58 92.31	62.06 70.51 76.31	84.34 88.19 93.79	22.3

Table 8.19. Experimental results of the macro-average dataset clustering experiments (Continued)

Extract Flat Clustering From Hierarchy + AGNES + Average Linkage + Squared Euclidean Distance, Minclusters	14.03 21.7 31.87	86.51 89.12 92.08	12.98 20.81 31.14	86.41 89.04 92.03	7.7 12.65 19.89	82.22 85.47 89.78	74.11 78.23 83.21	90.04 93.12 96.2	22.2
Extract Flat Clustering From Hierarchy + AGNES + Centroid + Squared Euclidean Distance, Minclusters	1.96 2.32 2.67	2.31 6.62 84.48	0.74 1.05 1.35	1.08 5.42 84.38	1 1.18 1.36	1.17 5.3 85.22	65.82 69.34 73.67	76.62 80.96 86.2	22.3
Extract Flat Clustering From Hierarchy + AGNES + Wards Linkage + Squared Euclidean Distance, Minclusters	63.55 68.05 74.34	72.39 77.12 82.36	63.32 67.85 74.19	72.23 76.99 82.27	65.54 71.44 79.01	89.9 92.46 94.84	58.73 65.05 70.99	60.21 66.17 73.49	22.1
K-Means Lloyd (Lloyd, 1982), K	12.34 22.95 39.5	54.71 61.08 69.22	11.37 22.15 38.97	54.44 60.84 69.04	7.04 14.95 30.12	59.07 66.91 75.52	45.22 53.17 61.14	49.01 56.25 64.26	37.4
K-Means Sort (Phillips, 2002), K	12.85 20.23 35.15	54.96 61.08 68.23	11.85 19.38 34.56	54.7 60.85 68.04	7.25 12.63 25.62	55.79 67.17 75.86	46.32 53.55 63.21	48.87 56.07 64.88	24
K-Means Compare (Phillips, 2002), K	15.42 23.08 39.48	55.19 61.84 67.93	14.46 22.28 38.94	54.93 61.61 67.74	9.01 14.98 29.49	60.54 67.6 74.8	46.61 52.78 62.18	49.24 57.05 64.26	21.3
K-Means Hamerly (Hamerly, 2010), K	28.73 42.55 53.86	57.8 62.59 69.55	28.02 42.08 53.54	57.55 62.36 69.39	19.27 35.36 51	58.61 67.74 74.7	47.13 54.66 61.48	51.47 58.23 66.75	29
K-Means Elkan (Elkan, 2003), K	26.84 41.47 56.13	55.77 62.67 70.9	26.12 40.98 55.83	55.5 62.44 70.72	18.13 33.96 53.55	59.64 67.9 75.06	48.93 54.91 62.35	50.93 58.24 67.18	7.8
K-Medians Lloyd (Bradley, et al., 1997), K	9.79 16.37 26.79	34.88 50.09 64.54	8.75 15.44 26.07	34.32 49.72 64.31	5.49 9.9 17.92	25.81 45.27 65.81	43.49 50.48 59.92	50.51 57.15 65.93	52.3
K-Means MacQueen (MacQueen, 1967), K	8.97 14.19 23.67	7.3 11.75 17.11	7.87 13.2 22.84	6.25 10.7 16.13	4.91 8.15 14.61	3.89 6.53 9.83	50 56.91 64.75	54.41 60.23 66.19	23.2
K-Medoids PAM (Kaufman & Rousseeuw, 1987), K	33.16 43.87 55.29	67.01 71.98 76.82	32.56 43.42 54.98	66.83 71.83 76.7	24.79 36.5 53.35	85.49 88.23 90.99	50.08 55.56 61.99	54.84 60.82 68.46	24.9
K-Medoids EM (Schubert, et al., 2015), K	32.18 48.26 63.96	81.77 85.87 89.4	31.5 47.81 63.69	81.65 85.78 89.33	20.9 37.07 57.36	80.92 85.93 91.7	65.15 71.49 77.54	78.86 85.88 91.14	58.9
K-Means Batched Lloyd, K	9.01 14.38 23.39	11.49 19.23 28.72	7.84 13.4 22.56	10.37 18.33 27.99	4.96 8.29 14.33	6.4 11.47 18.86	47.48 56 63.58	52.46 61.33 69.05	26.1

When the results in Table 8.19 are interpreted, our proposed framework clearly outperforms all of the previously proposed state-of-the-art systems. However, our algorithm is also the third slowest one even though we multi-task the distance calculation process between the candidates and the new record. After analyzing the run times of the all tasks by using Microsoft Visual Studio profiler, we have noticed that the most time taking task is the one that assign numerical weights to the tokens in our proposed system. Since our system works in incremental mode, in each turn of clustering, we have to re-assign weights to the compared records set. But, because other algorithms work in batch mode, they are provided with the token set that numerical weights are already assigned. Once we ignore the run time during the calculation of the weights, the run time of our algorithm reduces to 10.2 seconds from 49.6 seconds and becomes the second fastest. We plan to improve the performance of weight assignment task as a future study.

In Table 8.19, we see that AGNES + Average linkage performs best among to the other algorithms. It achieves 89.12% average F-measure which is the closest to our results. However, we have to note that the other algorithms have several very important advantages. They are provided with the entire dataset, unlike our method. Our system is incremental thus it clusters each record one by one. Additionally, they are provided with the exact number of the clusters. Further analysis reveals that the TF*IDF weighting system clearly performs much better than the Raw term frequency weighting scheme. One of the primary reason of this is since the entire dataset is provided, the TF*IDF weighting scheme is much better able to tokenize the records. However, unlike the previously proposed systems, our system is still able to display significant performance when the raw term frequency is used. Because our proposed matching system significantly prevents incorrect-merges.

To measure resistance of our system against overfitting problem, we have conducted the second experiments on a different dataset. This time, we have

selected 500 unique products from our price having products dataset randomly with a random order as the initial training dataset. Then from the remaining 1,221 price having products, we have generated another 100 macro-average datasets like before. This time, each macro-average dataset is only composed of 500 unique products.

On this new initial dataset, we have re-trained the entire system. With our previous best configuration, this new initial dataset performs 96.62 F-measure. After the re-training, the F-measure has increased to 98.10 with the following parameter changes that are shown in Table 8.20. Then with the new obtained best configuration, we have experimented on the newly composed macro-average dataset. The critical aspect of this approach is that the system first time sees the new macro-average dataset except for the previously generated static variables by domain experts such as brands list. While the previous best configuration performs average 95.46 F-measure on this new macro-average dataset, the newly trained configuration performs 95.01 F-measure average as shown in Table 8.21.

Table 8.20. Changed configurations after re-training on the dataset of new clustering experiments

Parameter Name	Old Best Value	New Best Value
Distance Metric	Squared Euclidean Distance	Tanimoto Non-Binary Distance
IDF Variant	IDF Smooth	BM25 IDF
TF Variant	BM25 TF	Raw Frequency
Max Distance to be Paired Threshold	0.61	0.79
<i>editDistanceProductCode</i>	2	3
<i>minSubstringRatio</i>	10%	30%
<i>minLCSRatioWordLenght</i>	8	7
Maximum Price Ratio	1.50	1.59

Table 8.21. Results of new clustering experiments

Configuration	F-measure	ARI	Precision	Recall
Previous Best	0.9216	0.9211	0.9778	0.8658
	0.9546	0.9543	0.9924	0.9197
	0.9730	0.9728	1.0	0.9480
New Best	0.8861	0.8853	0.8528	0.8947
	0.9501	0.9498	0.9645	0.9365
	0.9801	0.9800	1.0	0.9758

When we analyze the results in Table 8.21, we see that the results are very close. As expected, the previous best configuration performs better than the new configuration because the previous configuration was trained on the entire 1721 records while the new configuration is only trained on 500 records. These results clearly indicate that our proposed system is very robust and strong against overfitting problem.

8.4. Product Search Engine Performance Experiments

The performance of the proposed search engine completely depends on the success of the underlying classifier and clustering systems. The accuracy of the classifier system determines the performance of top category products ranking system. The success of the clustering system determines both the precision and recall of the product searching system.

We have conducted experiments as two different approaches. In the first approach, we have evaluated how well the existing commercial CSA systems performs in terms of precision. In the second approach, we have evaluated the best category of a product search by using public search engine Google.

8.4.1. Precision Experiments of Existing CSA Systems

We have evaluated the top 7 commercial CSA websites in Turkey with 5 different queries. For calculating precision, we have checked whether the top 10 returned results are related to the query or not. The results of precision experiments are presented in Table 8.22 where N/A means that those websites did not return any results.

Table 8.22. Precision experiments of top CSA services in Turkey

CSA Website	Query	Precision
www.akakce.com	nikon d810	100%
	dishonored pc oyun	100%
	canon cli-551	100%
	asus xonar dsx 7.1	100%
	fakir buharlı ütü	100%
www.cimri.com	nikon d810	100%
	dishonored pc oyun	N/A
	canon cli-551	100%
	asus xonar dsx 7.1	N/A
	fakir buharlı ütü	100%
www.teknofiyat.com	nikon d810	100%
	dishonored pc oyun	100%
	canon cli-551	100%
	asus xonar dsx 7.1	100%
	fakir buharlı ütü	N/A

Table 8.22. Precision experiments of top CSA services in Turkey (Continued)

www.iyi.net	nikon d810	N/A
	dishonored pc oyun	N/A
	canon cli-551	N/A
	asus xonar dsx 7.1	N/A
	fakir buharlı ütü	N/A
www.uygunfiyat.com	nikon d810	N/A
	dishonored pc oyun	N/A
	canon cli-551	N/A
	asus xonar dsx 7.1	N/A
	fakir buharlı ütü	N/A
www.efiyat.com	nikon d810	N/A
	dishonored pc oyun	N/A
	canon cli-551	N/A
	asus xonar dsx 7.1	N/A
	fakir buharlı ütü	N/A
www.fiyatarat.com	nikon d810	50%
	dishonored pc oyun	0%
	canon cli-551	100%
	asus xonar dsx 7.1	33%
	fakir buharlı ütü	0%
Our System	nikon d810	100%
	dishonored pc oyun	100%
	canon cli-551	100%
	asus xonar dsx 7.1	100%
	fakir buharlı ütü	100%

When the results in Table 8.22 are analyzed, we see that only our system and Akakce achieves 100% precision.

8.4.2. Performance Experiments of Top Products in a Category Search

For analyzing performance of top products in a category feature of our system, we have queried our system and Google and analyzed the results. For

example for the query “en iyi ekran kartı” (best graphic card), the top result in the Google is sponsored results. Basically they are the ads promoted by the E-commerce websites like shown in Figure 8.8. Moreover, two results of the displayed six results are not even graphic cards. The two results of the displayed four graphic cards are professional graphic cards and most of the users who look for graphic cards would not be interested in them. CSA systems cannot be queried with such “best of something” queries. However, CSA systems like AkAkce support categorical listing and in these listings, they sort the results by their internal ranking mechanisms such as in our search engine as shown in Figure 7.5.



Figure 8.8. The top result of Google “en iyi ekran kartı” (best graphic card) search

The 2-th, 10-th and 11-th results of Google lead to category listing of the E-commerce stores in Turkey like shown in Figure 8.9. It can be assumed as useful however, how they rank their sold graphic cards is a mystery. The 3-th, 6-th and 8-th results are journal posts about best graphic cards however, they are heavily outdated. The 4-th result is a video about best graphic cards and again it is outdated as well. The 5-th result is pictures of graphic cards. The 7-th and 9-th results of Google are about what consumers should consider when purchasing a graphic card. However, when a user query our system to find out best products in a category, our system always provide the most updated information and rank them according to our proposed ranking mechanism like shown in Figure 7.5.

Ekran Kartları Nvidia ve Atı Ekran Kartı Modelleri - Vatan Bilgisayar
www.vatanbilgisayar.com/ekran-kartlari/ ▼
 Nvidia ve ati gibi markalardan oluşan ekran kartı modellerini en uygun fiyatlarla ekran kartları ve özel taksit seçenekleriyle Vatan Bilgisayar'da
 Nvidia - Amd - ASUS Ekran Kartları - MSI Ekran Kartları

En İyi 10 Ekran Kartı - ShiftDelete.Net
shiftdelete.net/en-iyi-10-ekran-karti-20709/ ▼
 18 May 2010 - Piyasada bulunan onlarca ekran kartı modeli arasında en iyileri hangisi biliyor musunuz? Sizler için şu an satılan kartlar arasın da en iyi 10 ...

Dünyanın En Güçlü 10 Ekran Kartı - YouTube
<https://www.youtube.com/watch?v=2aiHBB6EDnE> ▼
 10 Şub 2015 - Donanım Arşivi tarafından yüklendi
 AMD'nin yeni sunucularında çok iyi bir şekilde güncellendi ... 3000 ti ekran kartı gta 5 için alacağım amegim en ...

en iyi ekran kartı ile ilgili görseller Görseller hakkında kötüye kullanım bildirin



en iyi ekran kartı için diğer resimler

Ekran kartı tavsiyesi 2015 Oyun içi En iyi ekran kartları
www.uzmantavsiyesi.net • Teknoloji ▼
 19 Nis 2015 - Ekran kartı tavsiyesi 2015 en kaliteli en ucuz en iyi hızlı pci oyun için popüler ekran kartı önerisi tavsiyeleri hangisini almalıyım hızlı fps ...

Ekran kartı seçiminde hedefi 12'den vurun! - Sayfa 3 - CHIP Online
www.chip.com.tr • Satın Alma Rehberi ▼
 Ekran kartı seçiminde AMD veya Nvidia mı sorusunu yanıtlamamız elbette ... Linux, size Windows veya OS X'i bırakılabilecek kadar iyi bir seviyeye geldi mi?

Nvidia ve AMD ekran kartları karşı karşıya! - CHIP Online
www.chip.com.tr • Galeri ▼
 Puan: 30,6 (100 üzerinden) Fiyat: 232 TL Sessizlik: %67,9 Ekipman: %62,0 Performans: %14,4 Güç tasarrufu: %48,8 Yüksek yüzdelere, o daha iyi olduğuna ...

Ekran Kartı Alırken Nelere Dikkat Edilmelidir? - Webtekno
www.webtekno.com/.../ekran-karti-alirken-nelere-dikkat-edilmelidir-h4106.html ▼
 8 Ara 2014 - (Boyle dedik diye Bit değeri yüksek olan benim ekran kartım daha iyi demesin. Çünkü buda ekran kartının daha iyi olmasında önemli olan ...

Ekran Kartı Modelleri ve Fiyatları - Gold.com.tr
www.gold.com.tr • Bilgisayar • Bilgisayar Bileşenleri ▼
 En iyi ekran kartı markalarını Gold'da bulabilirsiniz. Asus ekran kartı ve diğer markaların ekran kartı fiyatları çeşitli kampanyalarla her cebe uygun! Ekran kartı ...

Ekran Kartı Çeşitleri ve Fiyatları | Media Markt
www.mediamarkt.com.tr • Bilgisayar • Bilgisayar Bileşenleri • Ekran Kartı ▼
 Bilgisayarınızın performansına en uygun Ekran Kartı Çeşitlerine ... Bu konuda tercih yapılırken, en iyi

Figure 8.9. Google best graphic cards query

9. CONCLUSIONS AND FUTURE WORK

9.1. Focused Web Crawler Development

The methods we have presented in this study for the focused web crawler development process can significantly help .NET developers for any software development. According to our experiments, using .NET framework without fine tuning can significantly hinder the potential application performance and decrease the utilization of system resources. There can be even more performance tunings, however during EcCrawler development, source code compression, HTTP compression while fetching web pages, initialization of the thread pool, garbage collector mode setup and the maximum concurrent connection limit are the major ones that are discovered and applied. The proposed methods and experimental results presented in this study will help .NET developers to boost their software performance and save their time to figure the bottlenecks that can affect the expected performance of their application. For the future work, EcCrawler design and implementation may be modified to make it compatible for running on a cluster of computers rather than on a single machine. Additionally, a classifier may be embedded to the EcCrawler to determine product pages automatically.

9.2. Sentiment Analysis of E-Commerce Product Comments

In this study, we have presented through experimental results of sentiment analysis on different datasets and provided a detailed comparison of state-of-the-art weighting metrics. Our results clearly indicate that some of the previously proposed metrics need further experiments for verification. Furthermore, we propose that, with the proper parameter tuning, just using the statistical weighting schemes are as good as expensive linguistic methodologies. We observe that using log normalization TF * probabilistic IDF weighting scheme with L2 vector normalization has the best classification accuracy in both Turkish and English datasets. Additionally we provide preprocessed versions of all datasets that we

have experimented in this study on a GitHub repository⁵⁵. We believe that this contribution is significantly important since it provides easiness and objective comparability for the researchers. As a future study, we plan to do research for a better weighting scheme to employ in sentiment analysis task and perform more comprehensive experiments on the different datasets.

9.3. Record Linkage Task of E-Commerce Products

In this study, we have presented a novel record linkage system for E-commerce products. The system clearly outperforms the similar systems that are previously proposed. Additionally, our study is unique when all of its attributes are taken into consideration as far as to our knowledge. The product code matching system along with the other matching systems that we have proposed significantly improve the success rate of the record linkage task. Moreover, we have systematically experimented with the many different state-of-the-art metrics, and we have shown which one of them are promising in the record linkage task of E-commerce products. Furthermore, as far as to our knowledge, this study is the first one publishing a public E-commerce product dataset⁵⁶ in this area. As for the future work, the automatic generation of the lists of brand words and the conflicting words can be researched. Additionally, we have not conducted all combination of all of the experimented metrics. Thus, there is still space for improvement by finding a better combination.

⁵⁵ <https://github.com/FurkanGozukara/Sentiment-Analysis>

⁵⁶ <https://github.com/FurkanGozukara/Record-Linkage>

9.4. Proposed CSA System

In this thesis study, we have presented a full scale CSA system with covering every part of the system and we have provided possible solutions to encountered problems during the entire system development. We believe that, our proposed systems, approaches and published public datasets will be invaluable for the future researchers and the developers.





REFERENCES

- Abiteboul, S., Preda, M., & Cobena, G. (2003). Adaptive on-line page importance computation. Proceedings of the 12th international conference on World Wide Web, 280-290.
- Aggarwal, C. C., & Zhai, C. (2012). A survey of text clustering algorithms. Mining text data, 77-128.
- Aizawa, A., & Oyama, K. (2005). A fast linkage detection scheme for multi-source information integration. International Workshop on Challenges in Web Information Retrieval and Integration, 30-39.
- Akba, F., Uçan, A., Sezer, E. A., & Sever, H. (2014). Assessment of feature selection metrics for sentiment analyses: Turkish movie reviews. 8th European Conference on Data Mining 2014, 180-184.
- Arora, S., Mayfield, E., Penstein-Rosé, C., & Nyberg, E. (2010). Sentiment classification using automatically extracted subgraph features. Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, 131-139.
- Aytekin, Ç. (2013). An opinion mining task in Turkish language: A model for assigning opinions in Turkish blogs to the polarities. Journalism and Mass Communication, 3(3), 179-198.
- Baccianella, S., Esuli, A., & Sebastiani, F. (2009). Multi-facet rating of product review. European Conference on Information Retrieval, 461-472.
- Baeza-Yates, R., Castillo, C., Marin, M., & Rodriguez, A. (2005). Crawling a country: better strategies than breadth-first for web page ordering. Special interest tracks and posters of the 14th international conference on World Wide Web, 864-872.
- Bagga, A., & Baldwin, B. (1998). Entity-based cross-document coreferencing using the vector space model. Proceedings of the 17th international conference on Computational linguistics-Volume 1, 79-85.

- Bedi, P., Thukral, A., Banati, H., Behl, A., & Mendiratta, V. (2012). A multi-threaded semantic focused crawler. *Journal of Computer Science and Technology*, 27(6), 1233-1242.
- Berkhin, P. (2006). A survey of clustering data mining techniques. *Grouping multidimensional data*, Springer Berlin Heidelberg, 25-71.
- Bermingham, M. L., Pong-Wong, R., Spiliopoulou, A., Hayward, C., Rudan, I., Campbell, H., Wright, A. F., Wilson, J. F., Agakov, F., Navarro, P., & Haley, C. (2015). Application of high-dimensional feature selection: evaluation for genomic prediction in man. *Scientific reports*, 5, 1-12.
- Bhattacharya, I., & Getoor, L. (2004). Iterative record linkage for cleaning and integration. *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 11-18.
- Bhattacharya, I., & Getoor, L. (2006). A Latent Dirichlet Model for Unsupervised Entity Resolution. *SDM*, 5(7), 47-58.
- Bhattacharya, I., & Getoor, L. (2007). Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(5), 1-36.
- Bilenko, M., & Mooney, R. J. (2003a). Adaptive duplicate detection using learnable string similarity measures. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 39-48.
- Bilenko, M., & Mooney, R. J. (2003b). On evaluation and training-set construction for duplicate detection. *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 7-12.
- Bilenko, M., Basil, S., & Sahami, M. (2005). Adaptive product normalization: Using online learning for record linkage in comparison shopping. *Fifth IEEE International Conference on Data Mining (ICDM'05)*, 58-65.

- Bilenko, M., Kamath, B., & Mooney, R. J. (2006). Adaptive blocking: Learning to scale up record linkage. Sixth International Conference on Data Mining (ICDM'06), 87-96.
- BKM, B. K. (2016, 5 23). E-Commerce Transactions. Retrieved from Bankalararası Kart Merkezi (BKM): <http://bkm.com.tr/en/internetten-yapilan-kartli-odeme-islemleri>
- Blitzer, J., Dredze, M., & Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, 7, 440-447.
- Boldi, P., Codenotti, B., Santini, M., & Vigna, S. (2004). Ubicrawler: A scalable fully distributed web crawler. Software: Practice and Experience, 34(8), 711-726.
- Boynukalin, Z. (2012). Emotion analysis of turkish texts by using machine learning methods. M.S. Thesis, The Graduate School of Natural and Applied Sciences of Middle East Technical University, 69.
- Böhm, C., de Melo, G., Naumann, F., & Weikum, G. (2012). LINDA: distributed web-of-data-scale entity matching. Proceedings of the 21st ACM international conference on Information and knowledge management, 2104-2108.
- Bradley, P. S., Mangasarian, O. L., & Street, W. N. (1997). Clustering via concave minimization. Advances in neural information processing systems, 368-374.
- Brown, R. W. (1957). Linguistic determinism and the part of speech. The Journal of Abnormal and Social Psychology, 55(1), 1-5.
- Castillo, C. (2004). Effective Web Crawling. Ph.D. Thesis, Department of Computer Science, University of Chile, 179.
- Cavnar, W. B., & Trenkle, J. M. (1994). N-gram-based text categorization. Ann Arbor MI 48113-4001, 161-175.

- Chakrabarti, S., Van den Berg, M., & Dom, B. (1999). Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11), 1623-1640.
- Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), 39.
- Chaudhuri, S., Ganjam, K., Ganti, V., & Motwani, R. (2003). Robust and efficient fuzzy match for online data cleaning. *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 313-324.
- Chen, S. C., & Dhillon, G. S. (2003). Interpreting dimensions of consumer trust in e-commerce. *Information Technology and Management*, 4(2-3), 303-318.
- Chien, S., Dwork, C., Kumar, R., Simon, D. R., & Sivakumar, D. (2004). Link evolution: Analysis and algorithms. *Internet mathematics*, 1(3), 277-304.
- Cho, J., & Garcia-Molina, H. (2000). The Evolution of the Web and Implications for an Incremental Crawler. *Proceedings of the 26th International Conference on Very Large Data Bases*, 200-209.
- Cho, J., & Garcia-Molina, H. (2003). Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems (TODS)*, 28(4), 390-426.
- Cho, J., & Schonfeld, U. (2007). Rankmass crawler: a crawler with high personalized pagerank coverage guarantee. *Proceedings of the 33rd international conference on Very large data bases*, 375-386.
- Cho, J., Garcia-Molina, H., & Page, L. (1998). Efficient crawling through URL ordering. *Computer networks and ISDN systems*, 30(1-7), 161-172.
- Christen, P. (2008). Febrl: a freely available record linkage system with a graphical user interface. *Proceedings of the second Australasian workshop on Health data and knowledge management-Volume 80*, 17-25.

- Christen, P. (2012). A survey of indexing techniques for scalable record linkage and deduplication. *IEEE transactions on knowledge and data engineering*, 24(9), 1537-1555.
- Christen, P., Gayler, R., & Hawking, D. (2009). Similarity-aware indexing for real-time entity resolution. *Proceedings of the 18th ACM conference on Information and knowledge management*, 1565-1568.
- Cohen, W. W. (1998). Integration of heterogeneous databases without common domains using queries based on textual similarity. *ACM SIGMOD Record*, 27(2), 201-212.
- Cohen, W. W., & Richman, J. (2002). Learning to match and cluster large high-dimensional data sets for data integration. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 475-480.
- Cohen, W. W., Kautz, H., & McAllester, D. (2000). Hardening soft information sources. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 255-259.
- Cohen, W., Ravikumar, P., & Fienberg, S. (2003). A comparison of string metrics for matching names and records. *Kdd workshop on data cleaning and object consolidation*, 3, 73-78.
- Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, 1-8.
- Cui, X., Potok, T. E., & Palathingal, P. (2005). Document clustering using particle swarm optimization. *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005*, 185-191.
- Dahiwal, P., Raghuwanshi, M., & Malik, L. (2014). PDD Crawler: A focused web crawler using link and content analysis for relevance prediction. *arXiv preprint arXiv:1411.4366*, 245-253.

- Dave, K., Lawrence, S., & Pennock, D. M. (2003). Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. *Proceedings of the 12th international conference on World Wide Web*, 519-528.
- Defays, D. (1977). An efficient algorithm for a complete link method. *The Computer Journal*, 20(4), 364-366.
- Dehkharghani, R., Saygin, Y., Yanikoglu, B., & Oflazer, K. (2015). SentiTurkNet: a Turkish polarity lexicon for sentiment analysis. *Language Resources and Evaluation*, 1-19.
- Demirtas, E., & Pechenizkiy, M. (2013). Cross-lingual polarity detection with machine translation. *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining*, 9.
- Dong, X., Halevy, A., & Madhavan, J. (2005). Reference reconciliation in complex information spaces. *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 85-96.
- Doorenbos, R. B., Etzioni, O., & Weld, D. S. (1997). A scalable comparison-shopping agent for the world-wide web. *Proceedings of the first international conference on Autonomous agents*, 39-48.
- dos Santos, S. C., Angelim, S., & Meira, S. R. (2001). Building comparison-shopping brokers on the web. *Electronic Commerce*, 26-38.
- Edwards, J., McCurley, K., & Tomlin, J. (2001). An adaptive model for optimizing performance of an incremental web crawler. *Proceedings of the 10th international conference on World Wide Web*, 106-113.
- Elfeky, M. G., Verykios, V. S., & Elmagarmid, A. K. (2002). TAILOR: A record linkage toolbox. *Data Engineering, 2002. Proceedings. 18th International Conference on*, 17-28.
- Elkan, C. (2003). Using the triangle inequality to accelerate k-means. *ICML*, 3, 147-153.

- Elmagarmid, A. K., Ipeirotis, P. G., & Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, 19(1), 1-16.
- Erhan, C. (2015, 12 17). LibSVMsharp. (GitHub) Retrieved 6 23, 2016, from <https://github.com/ccerhan/LibSVMsharp>
- Eroğul, U. (2009). Sentiment Analysis in Turkish. M.S. Thesis, The Graduate School of Natural and Applied Sciences of Middle East Technical University, 55.
- Faloutsos, C., & Lin, K.-I. (1995). FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *SIGMOD '95 Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, 163-174.
- Fellegi, I. P., & Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64(328), 1183-1210.
- Florescuand, D. (2000). An extensible framework for data cleaning. *Proceedings of the 16th International Conference on Data Engineering*, 1.
- Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3(7-8), 1289-1305.
- Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *Science*, 315, 972-976.
- Fung, B. C., Wang, K., & Ester, M. (2003). Hierarchical document clustering using frequent itemsets. *SDM*, 3, 59-70.
- Gokhale, C., Das, S., Doan, A., Naughton, J. F., Rampalli, N., Shavlik, J., & Zhu, X. (2014). Corleone: hands-off crowdsourcing for entity matching. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 601-612.
- Gomes, D., & Silva, M. J. (2008). The Viúva Negra crawler: an experience report. *Software: practice & experience*, 38(2), 161-188.

- Google. (2016, 6 18). Fighting Spam. (Google Inside Search) Retrieved 6 18, 2016, from <https://www.google.com/insidesearch/howsearchworks/fighting-spam.html>
- Grannis, S. J., Overhage, J. M., Hui, S., & McDonald, C. J. (2003). Analysis of a Probabilistic Record Linkage Technique without Human Review. *AMIA Annual Symposium Proceedings*, 259-263.
- Gu, L., Baxter, R., Vickers, D., & Rainsford, C. (2003). Record linkage: Current practice and future directions. *CSIRO Mathematical and Information Sciences Technical Report*, 3, 32.
- Gupta, S. B. (2012). The Issues and Challenges with the Web Crawlers. *International Journal of Information Technology & Systems*, 1(1), 1-10.
- Guttman, R., Moukas, A., & Maes, P. (1999). Agents as mediators in electronic commerce. *Intelligent Information Agents*, 131-152.
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1), 10-18.
- Hall, R., & Fienberg, S. E. (2010). Privacy-preserving record linkage. *International conference on privacy in statistical databases*, 269-283.
- Hamerly, G. (2010). Making k-means even faster. *Proceedings of the 2010 SIAM International Conference on Data Mining*, 130-140.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2 ed.). California: Springer.
- Hernández, M. A., & Stolfo, S. J. (1995). The merge/purge problem for large databases. *ACM Sigmod Record*, 24(2), 127-138.
- Heydon, A., & Najork, M. (1999). Mercator: A scalable, extensible web crawler. *World Wide Web*, 2(4), 219-229.
- Hjaltason, G. R., & Samet, H. (1998). Incremental distance join algorithms for spatial databases. *ACM SIGMOD Record*, 27, 237-248.

- Hofmann, T., Schölkopf, B., & Smola, A. J. (2008). Kernel methods in machine learning. *The annals of statistics*, 1171-1220.
- Hotho, A., Staab, S., & Stumme, G. (2003). Ontologies improve text document clustering. *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, 541-544.
- Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2010). A practical guide to support vector classification. *Department of Computer Science National, Taiwan University, Taipei 106, Taiwan*, 1-16.
- Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 168-177.
- Huang, S.-L., & Tsai, Y.-H. (2011). Designing a cross-language comparison-shopping agent. *Decision Support Systems*, 50(2), 428-438.
- Inan, A., Kantarcioglu, M., Bertino, E., & Scannapieco, M. (2008). A hybrid approach to private record linkage. *IEEE 24th International Conference on Data Engineering*, 496-505.
- Isele, R., & Bizer, C. (2011). Learning linkage rules using genetic programming. *Proceedings of the 6th International Conference on Ontology Matching-Volume 814*, 13-24.
- Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. USA: Prentice-Hall, Inc.
- Jain, A., & Zongker, D. (1997). Feature selection: Evaluation, application, and small sample performance. *IEEE transactions on pattern analysis and machine intelligence*, 19(2), 153-158.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R (Vol. 6)*. New York: Springer.
- Jang, H., & Shin, H. (2010). Language-specific sentiment analysis in morphologically rich languages. *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, 498-506.

- Jin, L., Li, C., & Mehrotra, S. (2003). Efficient record linkage in large data sets. Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings. Eighth International Conference on, 137-146.
- Jones, K. S., Walker, S., & Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments: Part 1. Information Processing & Management, 36(6), 779-808.
- Kannan, A., Givoni, I. E., Agrawal, R., & Fuxman, A. (2011a). Matching unstructured product offers to structured product specifications. Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, 404-412.
- Kannan, A., Talukdar, P. P., Rasiwasia, N., & Ke, Q. (2011b). Improving product classification using images. IEEE 11th International Conference on Data Mining, 310-319.
- Karlsson, C. (2012, 8 28). Similarity and Dissimilarity Measures. (<http://inside.mines.edu>) Retrieved 6 22, 2016, from <http://web.simmons.edu/~benoit/lis642/Similarity%20and%20Dissimilarity%20Measures.pdf>
- Kassambara, A. (2015, 9 6). Hierarchical Clustering Essentials. (STHDA) Retrieved 6 23, 2016, from <http://www.sthda.com/english/wiki/hierarchical-clustering-essentials-unsupervised-machine-learning>
- Kaufman, L., & Rousseeuw, P. (1987). Statistical Data Analysis Based on the L1-Norm and Related Methods. Elsevier, North Holland: Amsterdam, 405-416.
- Kaufman, L., & Rousseeuw, P. J. (2008). Agglomerative Nesting (Program AGNES). Finding Groups in Data: An Introduction to Cluster Analysis, 199-252.

- Kaya, M., Fidan, G., & Toroslu, I. H. (2012). Sentiment analysis of turkish political news. Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology- Volume 01, 174-180.
- Kim, H.-s., & Lee, D. (2010). HARRA: fast iterative hashed record linkage for large-scale data collections. Proceedings of the 13th International Conference on Extending Database Technology, 525-536.
- Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. International Joint Conference on Artificial Intelligence, 14(2), 1137-1143.
- Koster, M. (2007). A Standard for Robot Exclusion. Retrieved 6 19, 2016, from The Web Robots Pages: <http://www.robotstxt.org/orig.html>
- Koudas, N., Sarawagi, S., & Srivastava, D. (2006). Record linkage: similarity measures and algorithms. Proceedings of the 2006 ACM SIGMOD international conference on Management of data, 802-803.
- Köpcke, H., & Rahm, E. (2010a). Frameworks for entity matching: A comparison. Data & Knowledge Engineering, 69(2), 197-210.
- Köpcke, H., Thor, A., & Rahm, E. (2009). Comparative evaluation of entity resolution approaches with fever. Proceedings of the VLDB Endowment, 2(2), 1574-1577.
- Köpcke, H., Thor, A., & Rahm, E. (2010b). Evaluation of entity resolution approaches on real-world match problems. Proceedings of the VLDB Endowment, 3(1-2), 484-493.
- Köpcke, H., Thor, A., Thomas, S., & Rahm, E. (2012). Tailoring entity resolution for matching product offers. Proceedings of the 15th International Conference on Extending Database Technology, 545-550.
- Krulwich, B. (1996). Information integration agents: BargainFinder and NewsFinder. Internet-Based Information Systems: Papers from the 1996 AAAI Workshop, 79.

- Kumar, M., & Vig, R. (2013). Focused crawling based upon tf-idf semantics and hub score learning. *Journal of Emerging Technologies in Web Intelligence*, 5(1), 70-77.
- Kushmerick, N. (1997). Wrapper induction for information extraction. Ph.D. Thesis, Department of Computer Science and Engineering, University of Washington, 245.
- Lan, M., Tan, C. L., Su, J., & Lu, Y. (2009). Supervised and traditional term weighting methods for automatic text categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4), 721-735.
- Lee, H.-K., Yu, Y.-H., Ghose, S., & Jo, G.-S. (2003). Comparison shopping systems based on semantic web--A case study of purchasing cameras. *Grid and Cooperative Computing*, 139-146.
- Li, G., & Liu, F. (2012). Application of a clustering method on sentiment analysis. *Journal of Information Science*, 38(2), 127-139.
- Li, X., Morie, P., & Roth, D. (2004). Robust reading: Identification and tracing of ambiguous names. *DTIC Document*, 8.
- Li, Y., Wang, Y., & Du, J. (2013). E-FFC: an enhanced form-focused crawler for domain-specific deep web databases. *Journal of Intelligent Information Systems*, 40(1), 159-184.
- Liu, B., & Zhang, L. (2012). A survey of opinion mining and sentiment analysis. *Mining text data*, 415-463.
- Liu, H., & Milios, E. (2012). Probabilistic models for focused web crawling. *Computational Intelligence*, 28(3), 289-328.
- Liu, W., & Du, Y. (2012). An improved topic-specific crawling approach based on semantic similarity vector space model. *Journal of Computational Information Systems*, 8, 8605-8612.
- Liu, Y., Li, Z., Xiong, H., Gao, X., & Wu, J. (2010). Understanding of internal clustering validation measures. *2010 IEEE International Conference on Data Mining*, 911-916.

- Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2), 129-137.
- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1(14), 281-297.
- Maimunah, S., Sastramihardja, H. S., Widyanoro, D. H., & Kuspriyanto, N. (2012). CT-FC: more Comprehensive Traversal Focused Crawler. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 10(1), 189-198.
- Manning, C. D., Raghavan, P., & Schütze, H. (2009). *An Introduction to Information Retrieval* Online edition. England: Cambridge University Press.
- Martineau, J., & Finin, T. (2009). Delta TFIDF: An Improved Feature Space for Sentiment Analysis. *Third AAAI International Conference on Weblogs and Social Media (ICWSM)*, 9, 106-110.
- McCallum, A., Nigam, K., & Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 169-178.
- McGlohon, M., Glance, N. S., & Reiter, Z. (2010). Star Quality: Aggregating Reviews to Rank Products and Merchants. *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, 114-121.
- Meilă, M. (2007). Comparing clusterings—an information based distance. *Journal of multivariate analysis*, 98(5), 873-895.
- Microsoft. (2016). .NET Framework 4.6 and 4.5. Retrieved 6 17, 2016, from Microsoft Developer Network (MSDN): <https://msdn.microsoft.com/en-us/library/w0x726c2%28v=vs.110%29.aspx>
- Microsoft. (2016). C#. Retrieved 6 17, 2016, from Microsoft Developer Network (MSDN): <https://msdn.microsoft.com/en-us/library/kx37x362.aspx>

- Microsoft. (2016). SQL Server 2014. Retrieved 6 17, 2016, from Microsoft TechNet: <https://technet.microsoft.com/en-us/library/hh278297%28v=sql.10%29.aspx>
- Microsoft. (2016). Welcome to Visual Studio 2013. Retrieved 6 17, 2016, from Microsoft Developer Network (MSDN): [https://msdn.microsoft.com/en-us/library/dd831853\(v=vs.120\).aspx](https://msdn.microsoft.com/en-us/library/dd831853(v=vs.120).aspx)
- Microsoft. (2016). Windows Presentation Foundation. Retrieved 6 17, 2016, from Microsoft Developer Network (MSDN): <https://msdn.microsoft.com/en-us/library/ms754130%28v=vs.110%29.aspx>
- Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38(11), 39-41.
- Monge, A., & Elkan, C. (1997). An efficient domain-independent algorithm for detecting approximately duplicate database records. *Proc of the SIGMOD*, 7.
- Montgomery, A. L., Hosanagar, K., Krishnan, R., & Clay, K. B. (2004). Designing a better shopbot. *Management Science*, 50(2), 189-206.
- Murtagh, F. (1983). A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4), 354-359.
- Najork, M., & Wiener, J. L. (2001). Breadth-first crawling yields high-quality pages. *Proceedings of the 10th international conference on World Wide Web*, 114-118.
- Nisha, J., & Sundareswari, K. (2015). Clustered Based User-Interest Ontology Construction for Selecting Seed URLs of Focused Crawler. *International Journal of Innovative Research in Computer and Communication Engineering*, 3(2), 827-830.
- O'Keefe, T., & Koprinska, I. (2009). Feature selection and weighting methods in sentiment analysis. *Proceedings of the 14th Australasian document computing symposium*, Sydney, 67-74.

- Olston, C., & Najork, M. (2010). Web Crawling. *Foundations and Trends in Information Retrieval*, 4(3), 175-246.
- Paltoglou, G., & Thelwall, M. (2010). A study of information retrieval weighting schemes for sentiment analysis. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 1386-1395.
- Pang, B., & Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, 8.
- Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *Proceedings of the 43rd annual meeting on association for computational linguistics*, 115-124.
- Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2), 1-135.
- Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, 10, 79-86.
- Pantel, P., & Lin, D. (199-206). Document clustering with committees. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 2002.
- Papavassiliou, V., Prokopidis, P., & Thurmair, G. (2013). A modular open-source focused crawler for mining monolingual and bilingual corpora from the web. *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, 43-51.
- Parlar, T. (2016). Feature Selection for Sentiment Analysis in Turkish Texts. Ph.D. Thesis, Çukurova University, Institute of Natural and Applied Sciences, The Faculty of Engineering and Architecture Electrical & Electronics Engineering.

- Pasula, H., Marthi, B., Milch, B., Russell, S., & Shpitser, I. (2002). Identity uncertainty and citation matching. *Advances in neural information processing systems*, 1401-1408.
- Pathak, B. (2010). A survey of the comparison shopping agent-based decision support systems. *Journal of Electronic Commerce Research*, 11(3), 178-192.
- Petrovski, P., Bryl, V., & Bizer, C. (2014). Integrating product data from websites offering microdata markup. *Proceedings of the 23rd International Conference on World Wide Web*, 1299-1304.
- Philip, S. (2010). Discussion of Similarity Metrics, Pearson Correlation Coefficient. Retrieved 6 22, 2016, from http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/sphilip/pear.html
- Phillips, S. J. (2002). Acceleration of k-means and related clustering algorithms. *Workshop on Algorithm Engineering and Experimentation*, 166-177.
- Pinkerton, B. (1994). Finding what people want: Experiences with the WebCrawler. *Proceedings of the Second International World Wide Web Conference*, 94, 17-20.
- Pivk, A., & Gams, M. (2002). Domain-dependent information gathering agent. *Expert systems with Applications*, 23(3), 207-218.
- Powers, D. M. (2007). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. School of Informatics and Engineering, Flinders University of South Australia.
- Prabowo, R., & Thelwall, M. (2009). Sentiment analysis: A combined approach. *Journal of Informetrics*, 3(2), 143-157.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1), 81-106.
- Rahm, E., & Do, H. H. (2000). Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4), 3-13.

- Rastogi, V., Dalvi, N., & Garofalakis, M. (2011). Large-scale collective entity matching. *Proceedings of the VLDB Endowment*, 4(4), 208-218.
- Reed, J. W., Jiao, Y., Potok, T. E., Klump, B. A., Elmore, M. T., & Hurson, A. R. (2006). TF-ICF: A new term weighting scheme for clustering dynamic data streams. *2006 5th International Conference on Machine Learning and Applications (ICMLA'06)*, 258-263.
- Rokach, L., & Maimon, O. (2005). Clustering methods. *Data mining and knowledge discovery handbook*, 321-352.
- Sadeddin, K. W., Serenko, A., & Hayes, J. (2007). Online shopping bots for electronic commerce: the comparison of functionality and performance. *International Journal of Electronic Business*, 5(6), 576-589.
- Sahoo, N., Callan, J., Krishnan, R., Duncan, G., & Padman, R. (2006). Incremental hierarchical clustering of text documents. *Proceedings of the 15th ACM international conference on Information and knowledge management*, 357-366.
- Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5), 513-523.
- Salton, G., Wong, A., & Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11), 613-620.
- Sarawagi, S., & Bhamidipaty, A. (2002). Interactive deduplication using active learning. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 269-278.
- Schubert, E., Koos, A., Emrich, T., Züfle, A., Schmid, K. A., & Zimek, A. (2015). A framework for clustering uncertain data. *Proceedings of the VLDB Endowment*, 8(12), 1976-1979.

- Schulz, J. (2008, 5 15). Hamming distance. (www.code10.info) Retrieved 6 22, 2016, from http://www.code10.info/index.php?option=com_content&view=article&id=59:hamming-distance&catid=38:cat_coding_algorithms_data-similarity&Itemid=57
- Seker, S. E., & Al-Naami, K. (2013). Sentimental analysis on Turkish blogs via ensemble classifier. Proc. the 2013 International Conference on Data Mining, 10-16.
- Shen, W., DeRose, P., Vu, L., Doan, A., & Ramakrishnan, R. (2007). Source-aware entity matching: a compositional approach. IEEE 23rd International Conference on Data Engineering, 196-205.
- Shen, W., Li, X., & Doan, A. (2005). Constraint-based entity matching. AAAI, 862-867.
- Shkapenyuk, V., & Suel, T. (2002). Design and implementation of a high-performance distributed web crawler. Data Engineering, 2002. Proceedings. 18th International Conference on, 357-368.
- Sibson, R. (1973). SLINK: an optimally efficient algorithm for the single-link cluster method. The computer journal, 16(1), 30-34.
- Singh, P. K., & Husain, M. S. (2014). Methodological study of opinion mining and sentiment analysis techniques. International Journal on Soft Computing, 5(1), 11-21.
- Singhal, A. (2001). Modern information retrieval: A brief overview. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 24(4), 35-43.
- Singla, P., & Domingos, P. (2005). Object identification with attribute-mediated dependences. European Conference on Principles of Data Mining and Knowledge Discovery, 297-308.
- Sokal, R., & Michener, C. (1938). A statistical method for evaluating systematic relationships. Univ. Kans. Sci. Bull., 38, 1409-1438.

- Souza, C. (2016, 6 1). BrayCurtis.cs. (Accord.NET on GitHub) Retrieved 6 22, 2016, from <https://github.com/accord-net/framework/blob/development/Sources/Accord.Math/Distances/BrayCurtis.cs>
- Sriram, B. (2010). Short text classification in twitter to improve information filtering. M.S. Thesis, Computer Science and Engineering, The Ohio State University, 69.
- Sutton, C. (2007, 10 22). Returns the KL divergence. (Univ. of Massachusetts Amherst, Computer Science Dept.) Retrieved 6 22, 2016, from <http://www.java2s.com/Code/Java/Development-Class/ReturnstheKLdivergenceKp1p2.htm>
- Team, O. D. (2016, 6 22). Introduction to Support Vector Machines. (OpenCV.org) Retrieved 6 22, 2016, from http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html
- Tejada, S., Knoblock, C. A., & Minton, S. (2002). Learning domain-independent string transformation weights for high accuracy object identification. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 350-359.
- Toutanova, K., & Moore, R. C. (2002). Pronunciation modeling for improved spelling correction. Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, 144-151.
- Uzun, E., Serdar Güner, E., Kılıçaslan, Y., Yerlikaya, T., & Agun, H. V. (2014). An effective and efficient Web content extractor for optimizing the crawling process. Software: Practice and Experience, 44(10), 1181-1199.
- Van Rijsbergen, C. (1979). Information Retrieval. Department of Computing Science, University of Glasgow, 112-113.
- Veress, G. (2013, 10 16). Clustering training. Retrieved 8 10, 2016, from <http://www.slideshare.net/gveress/cluster-training-2013>

- Veropoulos, K., Campbell, C., & Cristianini, N. (1999). Controlling the sensitivity of support vector machines. *Proceedings of the international joint conference on AI*, 55-60.
- Vinodhini, G., & Chandrasekaran, R. (2012). Sentiment analysis and opinion mining: a survey. *International Journal*, 2(6), 282-292.
- Wan, Y. (2009). *Comparison-Shopping Services and Agent Designs*. Newyork: IGI Global.
- Wells, J. (2000). Orthographic diacritics and multilingual computing. *Language problems & language planning*, 24(3), 249-272.
- Whang, S. E., Menestrina, D., Koutrika, G., Theobald, M., & Garcia-Molina, H. (2009). Entity resolution with iterative blocking. *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 219-232.
- Wiesner, A. (2016). Ward's Method. (Penn State Science) Retrieved 6 24, 2016, from http://sites.stat.psu.edu/~ajw13/stat505/fa06/19_cluster/09_cluster_wards.html
- Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. *Proceedings of the conference on human language technology and empirical methods in natural language processing*, 347-354.
- Wilson, T., Wiebe, J., & Hoffmann, P. (2009). Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational linguistics*, 35(3), 399-433.
- Winkler, W. E. (1995). Matching and record linkage. *Business survey methods*, 1, 355-384.
- Winkler, W. E. (2006). Overview of record linkage and current research directions. *Bureau of the Census*, 43.

- Yan, S., Lee, D., Kan, M.-Y., & Giles, L. C. (2007). Adaptive sorted neighborhood methods for efficient record linkage. *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, 185-194.
- Yang, J., Choi, J., Kim, J., Ham, H., & Lee, K. (2000a). A More Scalable Comparison Shopping Agent. In *Proc'EIS2000 Engineering of Intelligent Systems*, 7.
- Yang, J., Lee, E., & Choi, J. (2000b). A shopping agent that automatically constructs wrappers for semi-structured online vendors. *Intelligent Data Engineering and Automated Learning IDEAL 2000. Data Mining, Financial Engineering, and Intelligent Agents*, 368-373.
- Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. *ICML*, 97, 412-420.
- Yessenalina, A., Choi, Y., & Cardie, C. (2010). Automatically generating annotator rationales to improve sentiment classification. *Proceedings of the ACL 2010 Conference Short Papers*, 336-341.
- Yeung, K. Y., & Ruzzo, W. L. (2001). Details of the adjusted Rand index and clustering algorithms, supplement to the paper "An empirical study on principal component analysis for clustering gene expression data". *Bioinformatics*, 17(9), 763-774.
- Yıldırım, E., Çetin, F. S., Eryiğit, G., & Temel, T. (2015). The impact of NLP on Turkish sentiment analysis. *TÜRKİYE BİLİŞİM VAKFI BİLGİSAYAR BİLİMLERİ ve MÜHENDİSLİĞİ DERGİSİ*, 7(1).
- Yohanes, B. W., Handoko, H., & Wardana, H. K. (2011). Focused Crawler Optimization Using Genetic Algorithm. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 9(3), 403-410.
- Zhang, D., Rubinstein, B. I., & Gemmell, J. (2015). Principled graph matching algorithms for integrating multiple data sources. *IEEE Transactions on Knowledge and Data Engineering*, 27(10), 2784-2796.

- Zhang, K., Cheng, Y., Liao, W.-k., & Choudhary, A. (2011). Mining millions of reviews: a technique to rank products based on importance of reviews. Proceedings of the 13th International Conference on Electronic Commerce, 8.
- Zhang, R., & Tran, T. (2008). An entropy-based model for discovering the usefulness of online product reviews. Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01, 759-762.



CURRICULUM VITAE

Furkan GÖZÜKARA was born in Mersin in 1987. He has received his B.Sc. in Computer Engineering (CE), Istanbul Technical University (ITU)/Turkey in 2009. He has completed his M.Sc. in Computer Engineering, Mersin University/Turkey in 2012. His current research interests are sentiment analysis, record linkage, web crawler development, search engine development, web interface development, database management, windows server management, and building high-performance multi-threaded .NET applications.

APPENDIX

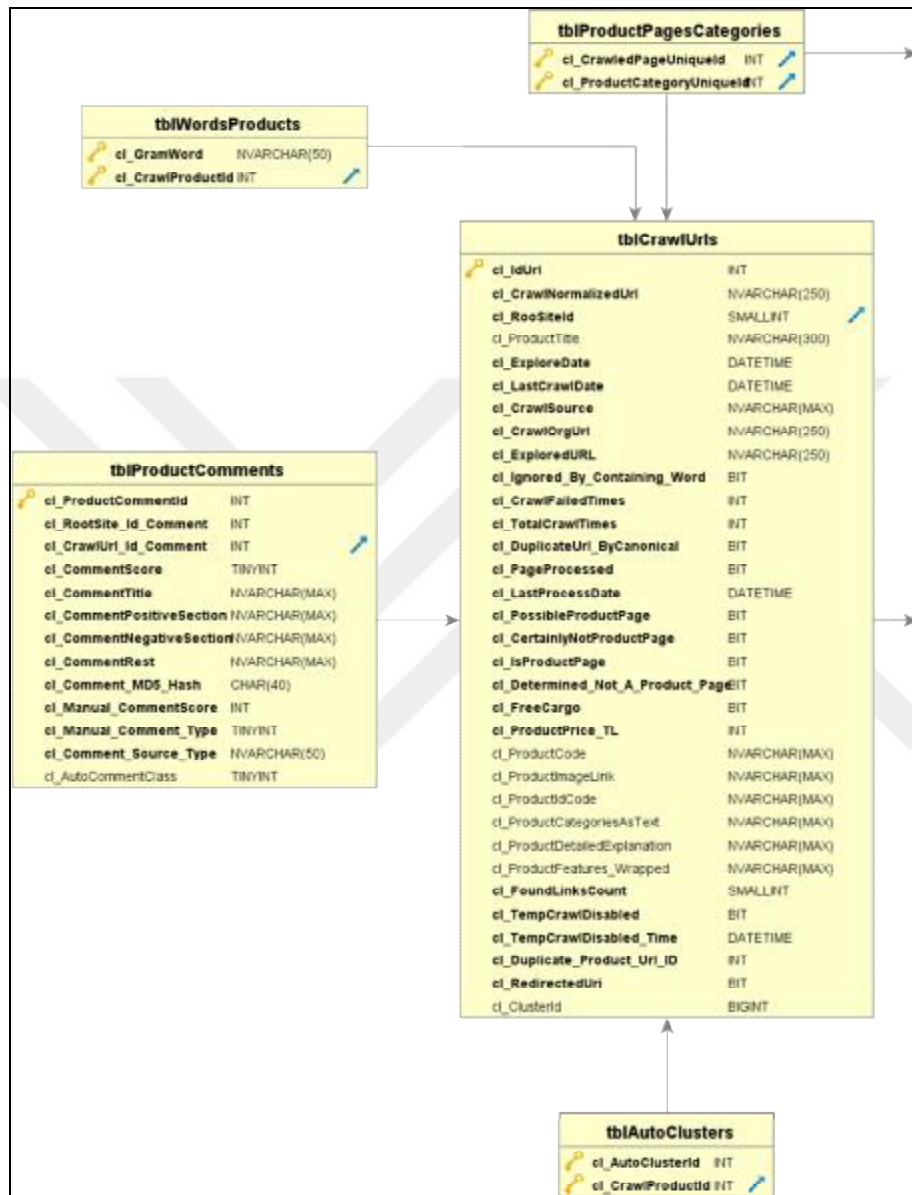


Figure 1. Details of tables in relational database schema of EcCrawler part 1

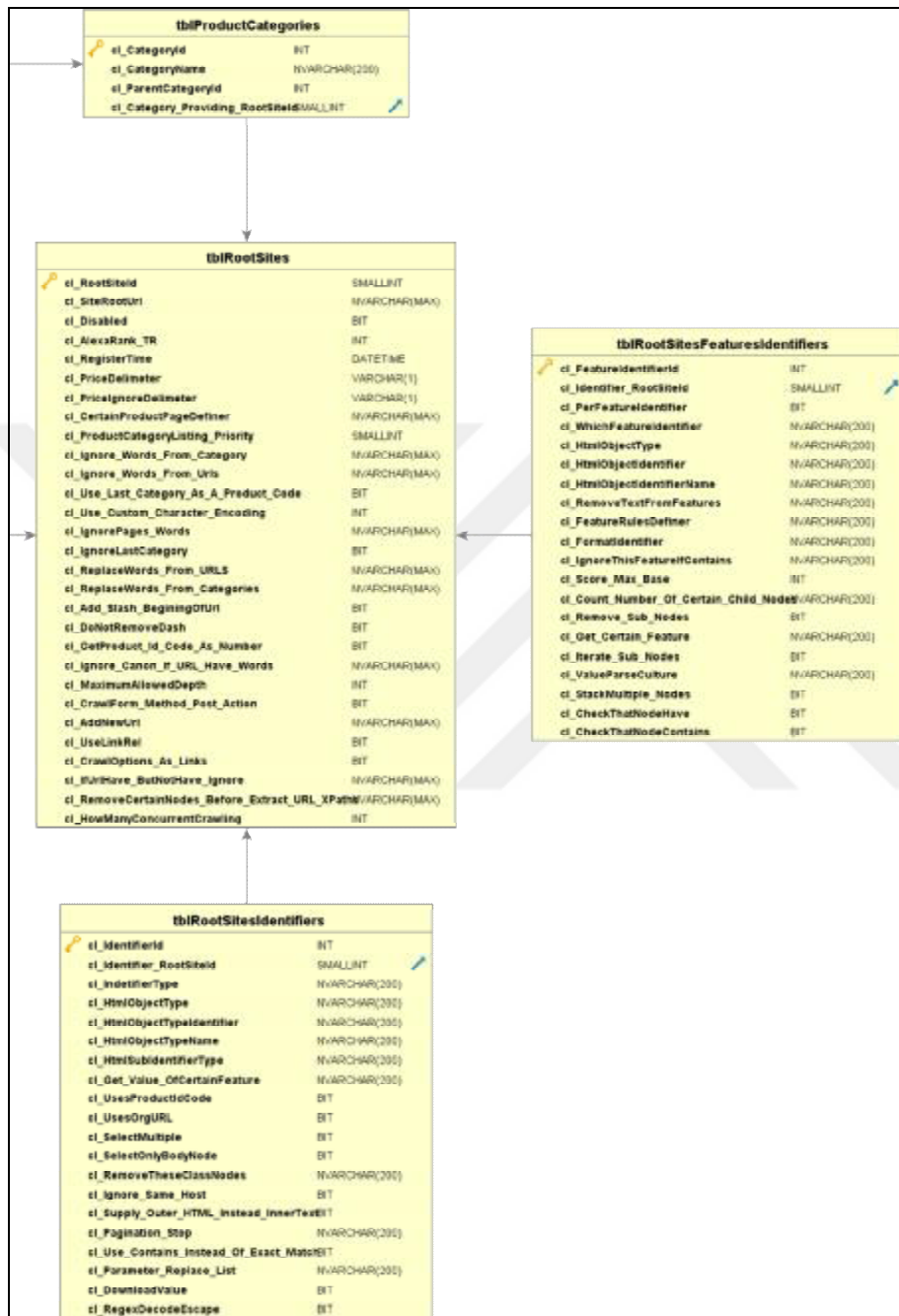


Figure 2. Details of tables in relational database schema of EcCrawler part 2

```
ThreadPool.SetMaxThreads(100000, 100000);
ThreadPool.SetMinThreads(100000, 100000);
```

Code Snippet 1. How to initialize the Thread Pool

```
AppDomain currentDomain = AppDomain.CurrentDomain; //set the event domain

Application.Current.DispatcherUnhandledException += new
//setup crashing event handler for sub threads
    DispatcherUnhandledExceptionEventHandler(
        CloseCrashHandlers.AppDispatcherUnhandledException);

currentDomain.UnhandledException += new
//setup crashing event handler for main thread
    UnhandledExceptionEventHandler(CloseCrashHandlers.CrashCloseHandler);

Closing += new CancelEventHandler(CloseCrashHandlers.CloseHandler);
//setup closing event handler
```

Code Snippet 2. How to setup global exception handler and close handlers

```
Task.Factory.StartNew(() =>
{
    StartTasks();
}, CancellationToken.None, TaskCreationOptions.LongRunning,
TaskScheduler.Default);
```

Code Snippet 3. How to start new threads independently from the main thread

```
private static Timer _timer;
private static int howManySeconds = 1;

public static void func_StartCrawlingWaitingUrls()
{
    _timer = new Timer(wrapper_func_CheckWaitingUrls, null,
        PublicSettings.irTimers_Delayed_Start_MiliSeconds,
        howManySeconds * 1000);
}
```

Code Snippet 4. How to start timer-based polling functions

```

private static System.Threading.Timer _timer;

_timer = new Timer(updateGlobalStatistics, null,
    PublicSettings.irTimers_Delayed_Start_MiliSeconds,
    PublicSettings.ir_RefreshUI_MS); //Start polling thread

object updateLock = new object();

private void updateGlobalStatistics(object sender)
{
    if (Monitor.TryEnter(updateLock))
    {
        try
        {
            Application.Current.Dispatcher.Invoke(new Action(() =>
            {
                //Get data from global static class and update the interface
            }));
        }
        finally
        {
            UiRefreshed();
            Monitor.Exit(updateLock);
        }
    }
}

```

Code Snippet 5. How to refresh the user interface without causing UI freeze

```

<configuration>
  <runtime>
    <gcAllowVeryLargeObjects enabled="true"/>
    <!--If the application uses over 2GB objects-->
    <gcServer enabled="true"/> <!--Set garbage collector mode-->
  </runtime>
</configuration>

```

Code Snippet 6. How to set Garbage Collector mode

```

private static readonly object _lockObject = new object();
//object to use for locking

public static void funcitonName(object inPut)
{
    lock (_lockObject)
    {
        //lock the object so non-thread safe objects are now thread-safe
        {
            //execute necessary procedures on non-thread safe objects such
            as dictionaries
        }
    }
}

```

Code Snippet 7. An example for usage of data structures and locking to ensure thread safety and data consistency

```

HttpRequest request = (HttpRequest)WebRequest.Create(srUrl);
//init request
request.KeepAlive = true; //keep connection alive
request.Accept =
    "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8";
//set request headers
WebHeaderCollection myWebHeaderCollection = request.Headers;
myWebHeaderCollection.Add("Accept-Language", "en-gb,en;q=0.5");
myWebHeaderCollection.Add("Accept-Encoding", "gzip, deflate");
//set accepting http compression

request.AutomaticDecompression = DecompressionMethods.Deflate |
    DecompressionMethods.GZip; //enable automatic decompression
using (WebResponse response = request.GetResponse()) //start response
{
    using (Stream strumien = response.GetResponseStream())
    {
        Encoding myEncoding; string srContentType = "";
        if (response.ContentType != null)
        {
            srContentType = response.ContentType;
            if (srContentType.Contains(";"))
            {
                srContentType = srContentType.Split(';')[1];
            }
            srContentType = srContentType.Replace("charset=", "");
            srContentType = //try to auto get encoding type
                PublicStaticFunctions.func_Process_Html_Input(srContentType);
        }
        try { myEncoding = Encoding.GetEncoding(srContentType); }
        catch { myEncoding = irCustomEncoding == 0 ? Encoding.UTF8 :
            Encoding.GetEncoding(irCustomEncoding); }
        //set recognized encoding
        using (StreamReader sr = new StreamReader(strumien, myEncoding))
        {
            srBody = sr.ReadToEnd(); //read the response
            srFinalUrl =
                PublicStaticFunctions.Return_Absolute_Url(
                    response.ResponseUri.AbsoluteUri.ToString(),
                    response.ResponseUri.AbsoluteUri.ToString());
        }
    }
}

```

Code Snippet 8. A shortened version of EcCrawler's fetching function

```
ServicePointManager.UseNagleAlgorithm = true;
//https://msdn.microsoft.com/en-
us/library/system.net.servicepointmanager.usenaglealgorithm
//Used to reduce network traffic by buffering small packets of data
// and transmitting them as a single packet

ServicePointManager.Expect100Continue = false;
//Setting this false will more likely to improve performance because
// some servers does not support it

ServicePointManager.CheckCertificateRevocationList = false;
//If certificate revocation is not important for developed crawler
// setting this false more likely to improve performance

ServicePointManager.DefaultConnectionLimit = 1000;
//This sets to maximum number of concurrent connections to same host
```

Code Snippet 9. How to increase the maximum number of concurrent connections to the same host and improve the fetching performance of the crawler by tuning the network configuration

```

public static string srConnectionString = "server=localhost;database=dbBame;
integrated security=SSPI; Max Pool Size=20000; Pooling=True;Connection
Timeout=3000;";//define connection string

public static bool batch_execute_CMD_update_delete(string srCommandText,
List<string> lst_Command_Variable_Names,
List<List<object>> lst_Command_Parameters, CommandType cdType)
//a generic function that can be called from any thread
{
    bool blErrorHappened = false;

    try
    {
        using (SqlConnection connection = //init SQL connection
            new SqlConnection(DbConnection.srConnectionString))
        {
            connection.Open(); //open SQL connection

            for (int i = 0; i < lst_Command_Parameters.Count; i++)
            {
                using (SqlCommand cmd =
                    new SqlCommand(srCommandText, connection))
                {
                    try //catch if any error happens
                    {
                        cmd.CommandTimeout = //set execution timeout
                            PublicSettings.irCommandTimeOutSettings_Second;
                        cmd.CommandType = cdType; //set command type

                        for (int k = 0;
                            k < lst_Command_Parameters[i].Count; k++)
                        {
                            //add command variables with an order
                            {
                                cmd.Parameters.AddWithValue(
                                    lst_Command_Variable_Names[k],
                                    lst_Command_Parameters[i][k]);
                            }

                            cmd.ExecuteNonQuery(); //execute SQL command
                        }
                    }
                    catch (Exception E) { //log errors here }
                }
            }
        }
    }
    catch (Exception E) { //log errors here}

    return !blErrorHappened;
}

```

Code Snippet 10. A shortened version of batch command SQL query execution function of EcCrawler

```
Interlocked.Increment(ref GlobalStats.long_GlobalPageFetchFailCount);  
//thread safe increment  
Interlocked.Read(ref GlobalStats.long_GlobalPageFetchFailCount);  
//thread safe read  
private static readonly object _lockObject = new object();  
//read only object to have lightweight locking  
  
private static void functionName ()  
{  
    lock (_lockObject) //ensures thread safety inside  
    {  
        //do operations here, they are thread safe  
    }  
}
```

Code Snippet 11. Examples for proper locking to provide thread safety



```

private Timer _timer;
private void Button_Click(object sender, RoutedEventArgs e)
{
    Task.Factory.StartNew(() => //start task as a background thread
    {
        startCheckingTimer();
    }, CancellationToken.None, TaskCreationOptions.LongRunning,
    TaskScheduler.Default);
    //TaskScheduler.Default ensures separate than main thread
}

private void startCheckingTimer()
{
    _timer = new Timer(doStuffParallel, null, 100, 1000);
    //constantly starts new separate than the main thread tasks
    //with a timed interval even if error happens in the started task
}

private List<Task> startedTaskList = new List<Task>(); //init task list
private int irMaximumNumberOfParallelTasks = 100; //set # of parallel tasks
private void doStuffParallel(Object state)
{
    startedTaskList.RemoveAll(tsk => tsk.Status == //remove completed tasks
    TaskStatus.RanToCompletion);
    for (int i = 0;
        i < irMaximumNumberOfParallelTasks - startedTaskList.Count; i++)
    {
        Task myTask = Task.Factory.StartNew(() =>
        {
            //do operation here
        }, CancellationToken.None, TaskCreationOptions.LongRunning,
        TaskScheduler.Default);

        myTask.ContinueWith(t => LogError("log error here"),
        TaskContinuationOptions.OnlyOnFaulted);
        //even if unhandled error happens it will be handled and
        // application continue to run
        startedTaskList.Add(myTask);
    }
}

```

Code Snippet 12. An example for multi-thread management system that ensures responsiveness of the UI