CODET: A NEW ALGORITHM FOR CONTAINMENT AND NEAR DUPLICATE DETECTION IN TEXT CORPORA

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Emre Varol

January, 2012

ABSTRACT

CODET: A NEW ALGORITHM FOR CONTAINMENT AND NEAR DUPLICATE DETECTION IN TEXT CORPORA

Emre Varol
M.S. in Computer Engineering
Supervisor: Prof. Dr. Cevdet Aykanat
January, 2012

In this thesis, we investigate containment detection, which is a generalized version of the well known near-duplicate detection problem concerning whether a document is a subset of another document. In text-based applications, there are three way of observing document containment: exact-duplicates, near-duplicates, or containments, where first two are the special cases of containment. To detect containments, we introduce CoDet, which is a novel algorithm that focuses particularly on containment problem. We also construct a test collection using a novel pooling technique, which enables us to make reliable judgments for the relative effectiveness of algorithms using limited human assessments. We compare its performance with four well-known near duplicate detection methods (DSC, full fingerprinting, I-Match, and SimHash) that are adapted to containment detection. Our algorithm is especially suitable for streaming news. It is also expandable to different domains. Experimental results show that CoDet mostly outperforms the other algorithms and produces remarkable results in detection of containments in text corpora.

Keywords: Corpus Tree, Document Containment, Near-Duplicate Detection, Similarity, Test Collection Preparation, Algorithm.

ÖZET

CODET: YAZILI DOKÜMANLARDA KAPSAMA VE BENZERLİK TESPİTİ İÇİN YENİ BİR ALGORİTMA

Emre Varol
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Cevdet Aykanat
Ocak, 2012

Bu tezde, birbirine benzer doküman tespitinin genelleştirilmiş versiyonu olan bir dokümanın içerdiği bilgilerin başka bir doküman tarafından içerilip içerilmediğini ortaya koyan kapsama tespiti konusu incelenmiştir. Yazılı dokümanlarda dokümanlarin birbirini kapsaması üc farkli şekilde karşımıza çıkmaktadır: ilk durum dokümanların tamamen aynı olması, ikinci durum dokümanların oldukça benzer olması, üçüncü ve ilk iki durumun daha geniş kapsamlı hali ise bir dokümanın diğerini içermesi. Kapsama tespiti için CoDet ismini verdiğimiz özellikle peşisıra gelmekte olan haberler için kullanışlı yeni bir algoritma önermekteyiz. Ayrıca havuzlama tekniği vasıtasıyla sınırlı insan yardımı kullnarak algoritmaların etkinliğini ve verimliliğini güvenilir bir şekilde ölçmemizi sağlayan bir test koleksiyonu oluşturduk. CoDet'in performansını oldukça benzer doküman tespitinde kullanilan ve alanında başarılı kabul edilen dört farklı algoritma (DSC, full fingerprinting, I-Match ve SimHash) ile karşılaştırdık. Deneysel çalışmalarımızdan edindiğimiz bulgulara göre CoDet genellikle alternatif algoritmalardan daha iyi sonuç vermekte ve yazılı dokümanlar üzerinde kapsama tespiti konusunda kaydadeğer sonuçlar üretmektedir.

Anahtar sözcükler: Cümle Ağacı, Doküman Kapsama, Yakın Kopya Tespiti, Benzerlik, Test Koleksiyonu Hazırlanması, Algoritma.

Acknowledgement

I would like to thank my Dr. Cevdet Aykanat and also Fazlı Can for their supports and guidance during my research. I would like to thank Dr. Uğur Güdükbay and Dr. Osman Abul for their constructive comments.

I would like to thank all my colleagues for spending their time to discuss some ideas.

I would like to especially thank Dr. B. Barla Cambazoğu for broadening my vision.

Furthermore, I am grateful to my colleagues Enver Kayaaslan, Seher Acer, F. Şükrü Torun, Kadir Akbudak, and S. Burak Sağlam.

Finally, I would like to thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) for financial support.

To vulnerable children of Africa...

Contents

List of Figures

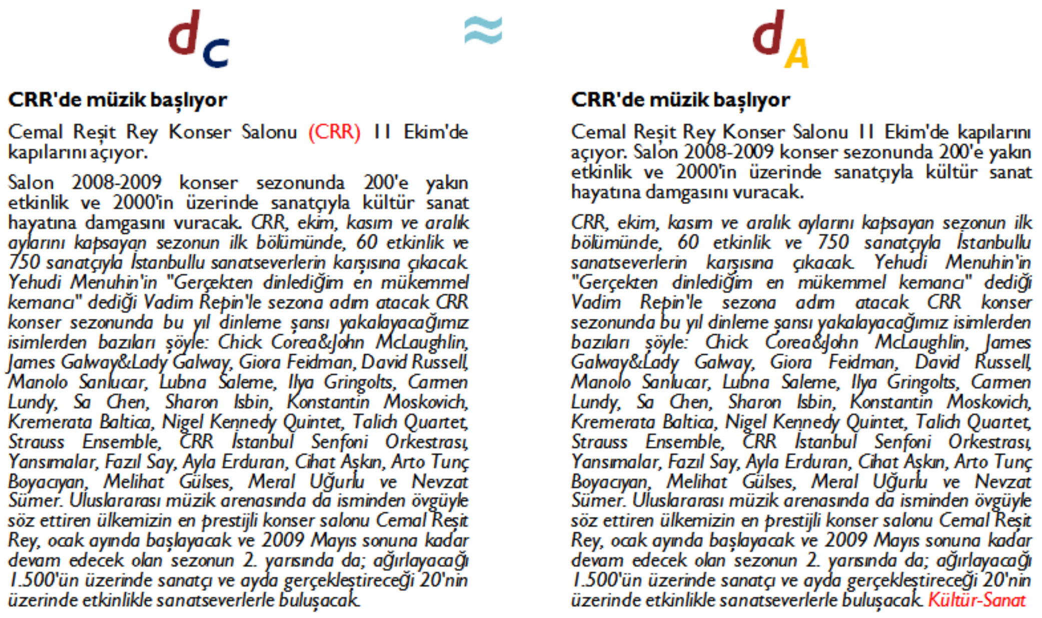
List of Tables

Chapter 1

Introduction

We consider a generalized version of the near-duplicate detection problem and investigate whether a document is a subset of another document [3]. In text-based applications, document containment can be observed in near-duplicates and containments. We refer to the problem of identifying such document pairs as the document containment detection problem. We study this problem within the context of news corpora that involve streaming news articles.

If a document d_C possesses all the text that document d_A has, then d_C is said to contain d_A, which is denoted as $d_C \supseteq d_A$, and this relation is called containment. Moreover, if two documents contain roughly the same content, they are near-duplicates [9]. Although near-duplicate condition is a special case of containment, these two cases are not usually distinguished from each other [22]. Similar to the conditional equivalence concept defined by Zobel and Bernstein [25], if $d_C \supseteq d_A$, then a news-consumer who have already read d_C would have no need to read d_A. Of course, $d_C \supseteq d_A$ does not necessarily imply $d_A \supseteq d_C$, i.e. containment relation is asymmetric. By detecting $d_C \supseteq d_A$, news consumers that have already seen d_C can be informed to skip reading d_A. We believe that it is more useful to detect containments rather than near-duplicates; because like a previously seen article, a reader does not want to read a new article that is a part of a already read one. An example of near-duplicate and containment cases is given in Fig. 1.1 and Fig. 1.2



CRR'de müzik başlıyor

Cemal Reşit Rey Konser Salonu (CRR) 11 Ekim'de kapılarını açıyor.

Salon 2008-2009 konser sezonunda 200'e yakın etkinlik ve 2000'in üzerinde sanatçıyla kültür sanat hayatına damgasını vuracak. *CRR, ekim, kasım ve aralık aylarını kapsayan sezonun ilk bölümünde, 60 etkinlik ve 750 sanatçıyla İstanbullu sanatseverlerin karşısına çıkacak. Yehudi Menuhin'in "Gerçekten dinlediğim en mükemmel kemancı" dediği Vadim Repin'le sezona adım atacak CRR konser sezonunda bu yıl dinleme şansı yakalayacağımız isimlerden bazıları şöyle: Chick Corea&John McLaughlin, James Galway&Lady Galway, Giora Feidman, David Russell, Manolo Sanlucar, Lubna Saleme, Ilya Gringolts, Carmen Lundy, Sa Chen, Sharon Isbin, Konstantin Moskovich, Kremerata Baltica, Nigel Kennedy Quintet, Talich Quartet, Strauss Ensemble, CRR İstanbul Senfoni Orkestrası, Yansımalar, Fazıl Say, Ayla Erduran, Cihat Aşkın, Arto Tunç Boyacıyan, Melihat Gülses, Meral Uğurlu ve Nevzat Sümer. Uluslararası müzik arenasında da isminden övgüyle söz ettiren ülkemizin en prestijli konser salonu Cemal Reşit Rey, ocak ayında başlayacak ve 2009 Mayıs sonuna kadar devam edecek olan sezonun 2. yarısında da; ağırlayacağı 1.500'ün üzerinde sanatçı ve ayda gerçeklestireceği 20'nin üzerinde etkinlikle sanatseverlerle buluşacak.*

CRR'de müzik başlıyor

Cemal Reşit Rey Konser Salonu 11 Ekim'de kapılarını açıyor. Salon 2008-2009 konser sezonunda 200'e yakın etkinlik ve 2000'in üzerinde sanatçıyla kültür sanat hayatına damgasını vuracak.

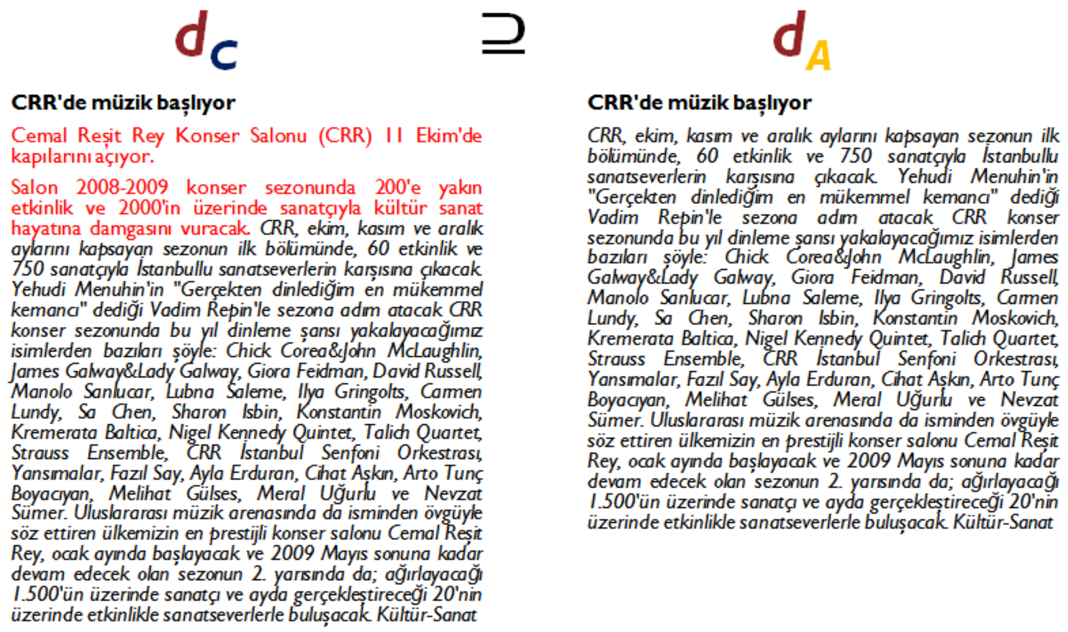
CRR, ekim, kasım ve aralık aylarını kapsayan sezonun ilk bölümünde, 60 etkinlik ve 750 sanatçıyla İstanbullu sanatseverlerin karşısına çıkacak. Yehudi Menuhin'in "Gerçekten dinlediğim en mükemmel kemancı" dediği Vadim Repin'le sezona adım atacak CRR konser sezonunda bu yıl dinleme şansı yakalayacağımız isimlerden bazıları şöyle: Chick Corea&John McLaughlin, James Galway&Lady Galway, Giora Feidman, David Russell, Manolo Sanlucar, Lubna Saleme, Ilya Gringolts, Carmen Lundy, Sa Chen, Sharon Isbin, Konstantin Moskovich, Kremerata Baltica, Nigel Kennedy Quintet, Talich Quartet, Strauss Ensemble, CRR İstanbul Senfoni Orkestrası, Yansımalar, Fazıl Say, Ayla Erduran, Cihat Aşkın, Arto Tunç Boyacıyan, Melihat Gülses, Meral Uğurlu ve Nevzat Sümer. Uluslararası müzik arenasında da isminden övgüyle söz ettiren ülkemizin en prestijli konser salonu Cemal Reşit Rey, ocak ayında başlayacak ve 2009 Mayıs sonuna kadar devam edecek olan sezonun 2. yarısında da; ağırlayacağı 1.500'ün üzerinde sanatçı ve ayda gerçeklestireceği 20'nin üzerinde etkinlikle sanatseverlerle buluşacak. Kültür-Sanat

Figure 1.1: Sample Near Duplicate Relation

In related studies, containment problem is addressed by near-duplicate detection algorithms [3]. Therefore, we compare performance of our proposed algorithm CoDet, whose name is originated from Containment Detection, with well-known near-duplicate detection approaches.

1.1 Motivation

Near-duplicate detection is an important task in various web applications. Due to reasons such as mirroring, plagiarism, and versioning such documents are common in many web applications [25]. One of the main reason behind is the rapid growth of the web, which eventually causes redundant information. Advances in the web technologies also increase the number of online news sources. These news sources generally disseminate slightly different versions of news stories coming from syndicated agencies by making small changes in the news articles. Identifying such documents increases the efficiency and effectiveness of search engines. It is also helpful for readers because most people tend to use more than one news source and nobody wants to read almost the same content twice. When comparing two



$d_C \supseteq d_A$

CRR'de müzik başlıyor

Cemal Reşit Rey Konser Salonu (CRR) 11 Ekim'de kapılarını açıyor.

Salon 2008-2009 konser sezonunda 200'e yakın etkinlik ve 2000'in üzerinde sanatçıyla kültür sanat hayatına damgasını vuracak. *CRR, ekim, kasım ve aralık aylarını kapsayan sezonun ilk bölümünde, 60 etkinlik ve 750 sanatçıyla İstanbullu sanatseverlerin karşısına çıkacak. Yehudi Menuhin'in "Gerçekten dinlediğim en mükemmel kemancı" dediği Vadim Repin'le sezona adım atacak CRR konser sezonunda bu yıl dinleme şansı yakalayacağımız isimlerden bazıları şöyle: Chick Corea&John McLaughlin, James Galway&Lady Galway, Giora Feidman, David Russell, Manolo Sanlucar, Lubna Saleme, Ilya Gringolts, Carmen Lundy, Sa Chen, Sharon Isbin, Konstantin Moskovich, Kremerata Baltica, Nigel Kennedy Quintet, Talich Quartet, Strauss Ensemble, CRR İstanbul Senfoni Orkestrası, Yansımalar, Fazıl Say, Ayla Erduran, Cihat Aşkın, Arto Tunç Boyacıyan, Melihat Gülses, Meral Uğurlu ve Nevzat Sümer. Uluslararası müzik arenasında da isminden övgüyle söz ettiren ülkemizin en prestijli konser salonu Cemal Reşit Rey, ocak ayında başlayacak ve 2009 Mayıs sonuna kadar devam edecek olan sezonun 2. yarısında da; ağırlayacağı 1.500'ün üzerinde sanatçı ve ayda gerçekleştireceği 20'nin üzerinde etkinlikle sanatseverlerle buluşacak. Kültür-Sanat*

CRR'de müzik başlıyor

CRR, ekim, kasım ve aralık aylarını kapsayan sezonun ilk bölümünde, 60 etkinlik ve 750 sanatçıyla İstanbullu sanatseverlerin karşısına çıkacak. Yehudi Menuhin'in "Gerçekten dinlediğim en mükemmel kemancı" dediği Vadim Repin'le sezona adım atacak CRR konser sezonunda bu yıl dinleme şansı yakalayacağımız isimlerden bazıları şöyle: Chick Corea&John McLaughlin, James Galway&Lady Galway, Giora Feidman, David Russell, Manolo Sanlucar, Lubna Saleme, Ilya Gringolts, Carmen Lundy, Sa Chen, Sharon Isbin, Konstantin Moskovich, Kremerata Baltica, Nigel Kennedy Quintet, Talich Quartet, Strauss Ensemble, CRR İstanbul Senfoni Orkestrası, Yansımalar, Fazıl Say, Ayla Erduran, Cihat Aşkın, Arto Tunç Boyacıyan, Melihat Gülses, Meral Uğurlu ve Nevzat Sümer. Uluslararası müzik arenasında da isminden övgüyle söz ettiren ülkemizin en prestijli konser salonu Cemal Reşit Rey, ocak ayında başlayacak ve 2009 Mayıs sonuna kadar devam edecek olan sezonun 2. yarısında da; ağırlayacağı 1.500'ün üzerinde sanatçı ve ayda gerçekleştireceği 20'nin üzerinde etkinlikle sanatseverlerle buluşacak. Kültür-Sanat

Figure 1.2: Sample Containment Relation

documents, in our case two news articles, it is easy to detect if they are completely identical; however, if there is only slight changes between them, the problem gets more difficult.

1.2 Contributions

Contributions of this thesis are the following.

- ◇ Introduction and implementation a sentence-based containment detection method, which is especially suitable for streaming news and also adaptable to different text-based problem domains.
- ◇ Implementation four well-known near-duplicate document detection algorithms' (DSC, full fingerprinting, I-Match, and SimHash) adapted versions for containment detection.
- ◇ We show that our approach generally outperforms these methods in terms

of efficiency and effectiveness.

◇ Construction of a test collection to serve as a benchmark using a novel pooling technique, which enables us to make reliable judgments for the relative effectiveness of algorithms using limited human assessments.

◇ Development of a software tool for testing document pairs for containment.

1.3 Outline

The rest of the thesis organized as follows. Chapter 2 gives necessary background information and summarizes the past and recent related work on near-duplicate and containment detection. Chapter 3 explains the proposed algorithm *CoDet* in detail and illustrates its processing principles on a running example. Chapter 4 describes the environment used in the experiments, and also gives detailed information about algorithms in comparison (DSC, full fingerprinting, I-Match, and SimHash) and ends with test collection preparation. In Chapter 5, we provide the results from our experimental evaluation. Chapter 6 discusses the future work that is left to be done and Chapter 7 concludes the thesis.

Chapter 2

Background and Related Work

In this chapter, we first give required background information about the utilized information retrieval techniques in this work. It is followed by an overview of the studies on near-duplicate document detection.

2.1 Background

In this section, we first present well-known information retrieval techniques that are applied after document parsing and just before similarity measure computation. Most common techniques are *stemming, eliminating stop words,* and *term weighting*. Cohen's Kappa measure [16] is also covered since it is used in test collection preparation.

2.1.1 Stemming

In information retrieval, stemming is the process of reducing words to their stem, which is their approximate root form. The stem does not have to be the same with its morphological root. It is mostly sufficient to map related words to the same stem, which may be different from the real root. Stemming is not a new topic in computer science, algorithms on it have been studied widely. As a result, there exists several types of stemming algorithms [15] [12] [1]. Can et al. [7] state that stemming has a significant effect on information retrieval in Turkish but CoDet and the other algorithms used in our work do not make any semantic analysis; hence, we consider two cases: no stemming and first-5, 5-prefix characters of each word.

- ◇ **No-Stemming (NS):** This option does not apply stemming on the words, i.e. identity function, and treat them as if they are roots.
- ◇ **Fixed Prefix Stemming:** This approach is a pseudo stemming technique. In this method, words are simply truncated and first-5 characters of each word is treated as its root. Words with less than 5 characters are not reduced at all. Can et al. [7] experimentally show that first-5 give the best performance in Turkish.

2.1.2 Eliminating Stop Words

Stop words are the words that are too common in documents and could have some side effect during the evaluation and similarity measures. Therefore, they are filtered out prior to processing documents to avoid their side effects and to shrink the documents. There is no standard stop word list for Turkish; thus, we construct our own by taking the most frequent words in our collection, which contains 180 words as in Table 2.1.

2.1.3 Term Weighting

Inverted document frequency (IDF) is a widely used word weighting technique in information retrieval. It measures the general importance of a word in a corpus. As can be seen in Equation 2.1 it is obtained for a word by dividing the total number of documents by number of documents that contain the word, then quotient's logarithm is taken.

Table 2.1: 180 Stop Words Automatically Generated From BilCol-2005

acaba, altı, ama, ancak, artık, asla, aslında, az, bana, bazen, bazı, bazıları, bazısı, belki, ben
beni, benim, beş, bile, bir, birçoğu, birçok, birçokları, biri, birisi, birkaç, birkaçı, birşey, birşeyi
biz, bize, bizi, bizim, böyle, böylece, bu, buna, bunda, bundan, bunu, bunun, burada, bütün
çoğu, çoğuna, çoğunu, çok, çünkü, da, daha, de, değil, demek, diğer, diğeri, diğerleri, diye
dokuz, dolayı, dört, elbette, fakat, falan, felan, filan, gene, gibi, hala, hangi, hangisi, hani
hatta, hem, henüz, hep, hepsi, hepsine, hepsini, her, herkes, herkese, herkesi, hiç, hiçbiri
hiçbirine, hiçbirini, için, içinde, iki, ile, ise, işte, kaç, kadar, kendi, kendine, kendini, ki, kim
kime, kimi, kimin, kimisi, madem, mı, mi, mu, mü, nasıl, ne, neden, nedir, nerde, nerede
nereden, nereye, nesi, neyse, niçin, niye, on, ona, ondan, onlar, onlara, onlardan, onların, onu
onun, orada, oysa, oysaki, öbürü, ön, önce, ötürü, öyle, rağmen, sana, sekiz, sen, senden, seni
senin, siz, sizden, size, sizi, sizin, son, sonra, şayet, şey, şeyden, şeye, şeyi, şeyler, şimdi, şöyle
şu, şuna, şunda, şundan, şunlar, şunu, şunun, tabi, tamam, tüm, tümü, üç, üzere, var, ve, ya
yahut, yani, yedi, yerine, yine, yoksa, zaten, zira

$$IDF(w) = log(\frac{|D|}{|\{d_i \in D : w \in d_i\}|}) \tag{2.1}$$

where D is the set of documents, d_i is a document from D and w is a word.

In the simplest form, term frequency (TF) is taken as term appearance count divided by total term count for a document (Equation 2.2).

$$TF(w, d) = \frac{|\{t_i \in d : t_i = w\}|}{|d|} \tag{2.2}$$

where d is document, t_i is a term in d and w is a word.

As an example, consider a document collection D containing $|D| = 10000$ documents in total. Let d_A and d_C be two documents that contain $|d_A| = 100$ and $|d_B| = 200$ words, respectively. According to the equations, IDF(w)=log(10000/100)=2, TF(w,d_A)=5/100=0.05 and TF(w,d_B)=20/200=0.1. Although d_C contains d_A, term weight of w would be different for them if TF-IDF would be used, which would degrade the quality of our containment similarity measure given in Equation 3.1.

Table 2.2: A Sample Contingency Table For Two Annotators (A and B) on 100 Documents

		A	A
		Containment	No Containment
B	Containment	40	10
B	No Containment	20	30

2.1.4 Cohen's Kappa Measure

Cohen's Kappa (κ) is a statistical measure of inter-annotator agreement for categorical data [16]. κ is more effective than only taking the agreements into account since it also considers agreements that are occurred by chance. It is calculated as

$$\kappa = \frac{P(a) - P(c)}{1 - P(c)} \tag{2.3}$$

where $P(a)$ represents probability of agreement and $P(c)$ represents probability of agreement by chance between annotators. $P(a)$ is a measured value after annotations. $P(c)$ is calculated by using the observed data to determine each annotator's randomly selecting each category. During the test collection preparation, annotators are asked to determine whether shown document pair is containment or not. Therefore, we have 2 categories. Let A and B two annotators that are assigned to classify 100 document pairs. Assume their annotation results are tabulated as in Table 2.2. Then, both annotators agreed on 40 containment cases and 30 no containment cases. Thus, $P(a) = \frac{30+40}{100} = 0.7$. A marked a pair as containment with 60% probability. B marked a pair as containment with 50% probability. Their random agreement probability on containment is $0.6 \times 0.5 = 0.3$. Their random agreement probability on no containment is $0.4 \times 0.5 = 0.2$. In total, $P(c) = 0.3 + 0.2 = 0.5$. Applying the equation,

$$\kappa = \frac{P(a) - P(c)}{1 - P(c)} = \frac{0.7 - 0.5}{1 - 0.5} = 0.4 \tag{2.4}$$

gives us a κ value of 0.4 implying fair agreement between annotators.

2.2 Related Work

In near-duplicate detection similarity measurement plays an important role [17]. By using similarity, two documents are defined as duplicates if their similarity or resemblance [4] exceeds a certain threshold value. Such approaches are applied to identify roughly the same documents, which have the same content except for slight modifications [2]. In comparisons, factors other than similarity may also play a role. Conrad and Schriber [10] after consulting librarians deem that two documents are duplicates if they have 80% overlap and 20 variations in length.

Similarity measures may use all words in documents in calculation. Instead of using each word, a sequence of them, *shingles*, may be used. In shingling approaches, if two documents have significant number of shingles in common, then they are considered as similar (near-duplicate). Well-known shingling techniques include for example COPS [2] and DSC (Digital Syntactic Clustering) [4]. COPS uses the sentences (or small units) to generate hash codes and stores them in a table to see if a document contains a sentence. Wang and Chang propose using the sequence of sentence lengths for near-duplicate detection and they evaluated different configurations of sentence-level and word-level algorithms [21].

Shingling and similarity approaches suffer from efficiency issues. As a result a new strategy emerged which is based on hashing of the whole document. I-Match [9] is a commonly known approach that uses this strategy. It filters terms based on collection statistics (*idf* values). Charikars Simhash [8] method is based on the idea of creating a hash by using document features (words, bigram, trigrams, etc.). It compares bit differences of these signatures to decide if two documents are near-duplicate or not. Yang and Callan [22] use clustering concepts for efficiency. While clustering documents they use additional information extracted from documents and structural relationships among document pairs. Deng and Rafiei [11] propose a new algorithm and data structure for eliminating duplicates in streaming environment. Their algorithm is based on the idea that SBF evicts the stale information to create space for more recent documents.

Hajishirzi et al. [13] propose an adaptable method for near duplicate detection by representing documents as real valued sparse k-gram vectors, where weights are learnt to optimize a similarity function. Zhang et al. [23] address the partial-duplicate detection problem by doing sentence level near-duplicate detection and sequence matching. Their algorithm generates a signature for each sentence and sentences that have the same signature are considered as near-duplicates. Theobald et al. [19] propose SpotSigs algorithm that combines stopword antecedents with short chains of adjacent content terms to create signatures. Monostori et al. [18] propose a new algorithm based on suffix tree structure to detect false matches caused by usage of hash functions. They also achieve to save the required information in a directed acyclic graph, which is less space consuming than suffix tree.

Chapter 3

CoDet Algorithm

CoDet is a novel sentence-based containment detection algorithm [20]. In this chapter, we first introduce containment similarity concept, which is the basis of similarity scoring in CoDet. Then, by using containment similarity concept, the way of similarity calculation is shown. We also illustrate the process of the algorithm on a running example and give pseudo code to make it clear. At the end, complexity analysis of the proposed algorithm is done in two extreme cases.

3.1 Containment Similarity Concept

CoDet is a novel sentence-based containment detection algorithm. It requires similarity measurements between document pairs. Unlike to its fingerprinting based alternatives, it also detects similar parts of document pairs. CoDet employs a new similarity measure called containment similarity (CS). It measures to what extent a document d_A is contained by another document d_C, which is defined as

$$CS(d_C, d_A) = \sum_{s_i \in S_C} \sum_{s_j \in S_A} cs(s_i, s_j), \tag{3.1}$$

where S_A and S_C denote the set of sentences in d_A and d_C, respectively. The function cs(s_i,s_j) indicates containment similarity between sentences s_i and s_j, which is calculated as

$$cs(s_i, s_j) = \sum_{k=1}^{len_{f(s_i,s_j)}} k \times idf(w_{f(s_i,s_j),k}), \tag{3.2}$$

where $f(s_i, s_j)$ denotes the word sequence representing the longest word prefix match of the sentences s_i and s_j, len_t is the length of the word sequence t, and $w_{t,k}$ stands for the k^{th} word in the word sequence t. For example, let s_1 be "John is happy." and s_2 be "John is sad." Then $f(s_1, s_2)$ is a word sequence $< John, is >$, $len_{f(s_1,s_2)}$ is 2 and $w_{f(s_1,s_2),1}$ is *John*.

As can be seen in Equation 3.2, containment similarity between two sentences grows significantly as their word prefix match gets longer. The containment similarity of a document to itself is referred to as self-containment similarity (SCS) and is shown in equation 3.3. Note that if d_C contains d_A, then CS(d_C,d_A) may be greater than SCS(d_A) since d_C may have extra similarities in sentences that does not belong to d_A.

$$SCS(d_A) = CS(d_A, d_A) = \sum_{s_i \in S_A} \sum_{s_j \in S_A} cs(s_i, s_j), \tag{3.3}$$

3.2 Containment Similarity Calculation

For efficient calculation of containment similarities, we utilize a prefix tree like data structure called *corpus tree*. The corpus tree begins with a virtual root node which contains a pointer list storing the locations of the children nodes in the next level. In addition to pointer list, nodes other than the root contain a label and a document list. The label represents the nodes term and the document list contains visiting document ids.

Let d_A denote a document with a set of sentences $S_A = \{s_1, s_2 \ldots s_n\}$. Processing of d_A involves processing all of its sentences. Insertion of s_i $(1 \leq i \leq n)$ to the corpus tree is performed as follows. First, words of s_i are sorted according to their *idf* values in descending order. Let $< w_1, w_2 \ldots w_m >$ denote the sequence

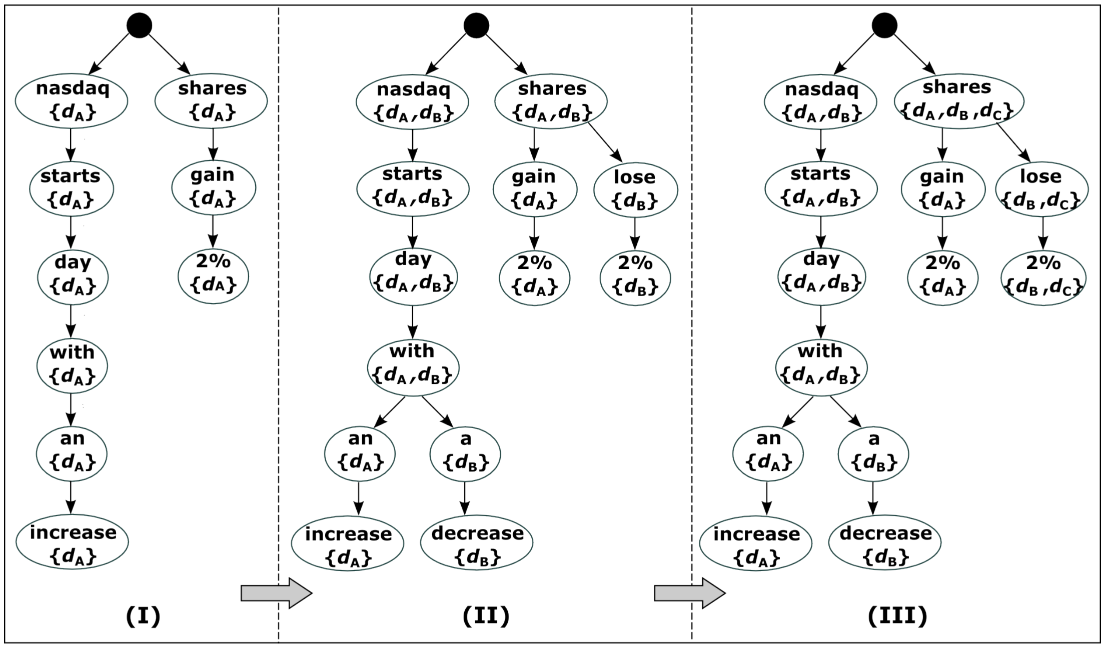


Figure 3.1: Insertion of three documents d_A:"NASDAQ starts day with an increase. Shares gain 2%.", d_B: "NASDAQ starts the day with a decrease. Shares lose 2%.", and d_C: "Shares lose 2%.". For the sake of clarity, words of sentences are not sorted according to their *idf* values. (Varol, E. Can, F., Aykanat, C., Kaya, O. "CoDet: Sentence-based Containment Detection in News Corpora", ACM CIKM '11 Conf., ©2011 ACM, Inc. http://dx.doi.org/10.1145/2063576.2063887. Reprinted by permission.)

of words in s_i after sorting. These words are inserted into the corpus tree starting from the virtual root node. If the root has a child ch_{w_1} with label w_1, then similarity values of d_A with all documents in ch_{w_1}'s visitor list are increased according to Equation 3.2. Otherwise, a new child node ch_{w_1} with label w_1 is created and added to the root's pointer list. In the next step, we apply the same operation on ch_{w_1} as we did the root, and insert the following word w_2 of s_i similarly. The insertion of s_i finishes after all of its words are processed. The remaining sentences of d_A are handled in the same manner. The same is performed on the remaining sentences of d_A.

Fig. 3.1 shows how the corpus tree grows with the sentence insertions. In Fig. 3.1-I, d_A's sentences "NASDAQ starts day with an increase."and "Shares gain 2%."are inserted to the corpus tree starting from the virtual root, which is indicated by a dark circle. Since the tree is initially empty, while inserting the first sentence all nodes with labels $< nasdaq, starts, day, with, an, increase >$ are created. Similarly, insertion of the second sentence creates nodes with labels $< shares, gain, 2\% >$. In Fig. 3.1-II, during the insertion of the sentence “NASDAQ starts day with a decrease.”previously created nodes with labels $< nasdaq, starts, day, with >$ are visited and updated. Also, two nodes with labels $< a, decrease >$ are created. Insertion of the sentence “Shares lose 2%.”visits the node with label *shares* and creates two nodes with labels $< lose, 2\% >$. Thus, similarity value of d_A and d_B is increased by summation of each revisited node’s impact value, which is calculated by multiplication of node’s depth and *idf* value of its label. For example, the contribution of the node with label *starts* is $2 \times log(\frac{3}{2}) + 1$ because its depth is 2 and word *starts* appears in 2 of 3 documents (in the experiments, the *idf* values are obtained from a large reference collection). The final structure of the corpus tree after the insertion of d_C is shown in Fig. 3.1-III.

To decide whether a document d_A is contained by another document d_C, CoDet uses $CS(d_A, d_C)$ as well as $SCS(d_A)$ values in Equation 3.1. If $\frac{CS(d_A,d_C)}{SCS(d_A)}$ exceeds the equivalency threshold level (ETL) parameter, then d_C is said to contain d_A. In the experiments, different ETL values are tested.

Algorithm 1 Parsing and Preprocessing Operations

```
Require: Document set $D = \{d_1, d_2 \ldots d_N\}$
 1: for each document $d_i \in D$ do
 2:     Parse $d_i$ and split into sentences
 3:     $S_i \leftarrow$ set of sentences in $d_i$
 4:     $scs[d_i] \leftarrow CalculateOwnScore(d_i)$
 5:     for each sentence $s_j \in S_i$ do
 6:         Split $s_j$ into words
 7:         $W \leftarrow$ set of words in $s_j$
 8:         Sort $W$ with respect to idf values of words
 9:         $InsertToCorpusTree(virtualRootNode, i, W, 0)$
10:     end for
11: end for
```

Algorithm 1 displays the parsing of a document and preprocessing operations on sentences just before their insertion into the corpus tree. These operations include splitting document into sentences, calculation of self containment similarity score (*scs*), splitting sentences into words, sorting words of a sentence according to *idf* values, and triggering *InsertToCorpusTree* function with *virtual root node*, document id (i), set of words of a sentence, and *depth*, whose value is zero since we start insertion operation from the root of the corpus tree. Note that the given algorithm seems to require a set of document; yet, only a few slight changes are needed to make it compatible with the online version of the problem where instead of a pile, documents come one by one.

Algorithm 2 *InsertToCorpusTree*(A node from the corpus tree *node*, integer document id i, A sentence from d_i *sentence*, integer *depth*)

```
1: if $node$ is $active$ then
2:    $node.visitorDocuments \leftarrow node.visitorDocuments \cup i$
3:    if $node.documentCount >$ document list size limit then
4:       Deactivate $node$
5:    else
6:       if $depth >$ depth threshold then
7:          return
8:       end if
9:       if $depth \neq 0$ then
10:         for Each document $d_j \in node.visitorDocuments$ do
11:            $IncreaseSimilarity(d_i, d_j, node.word, depth)$
12:         end for
13:      end if
14:   end if
15: end if
16: if $sentence$'s end is reached then
17:    return
18: end if
19: $nextWord \leftarrow$ next word of $sentence$
20: if $node$ has no child with label $nextWord$ then
21:    Generate a child to $node$ with label $nextWord$
22: end if
23: $InsertToCorpusTree(node.next(nextWord), i, sentence, depth + 1)$
```

Algorithm 2 shows how a document is inserted to the corpus tree in detail. In the experiments, we sometimes decide to deactivate nodes that are visited by too many documents since their effects on similarity measurement is negligible. For this reason, deactivation improves efficiency without degrading the effectiveness considerably. If the processing node is active the number of the processing document is added to its visitor list. Then, similarity values between the processing document and previous visitor documents are increased by *IncreaseSimilarity* function. If their similarity exceeds a threshold, it means a containment is detected and it is saved immediately. Details about similarity increase can be found in Algorithm 3. When the algorithm reaches to the last word of the processing sentence or depth threshold is exceeded, the process ends. Otherwise it continues with the next node. If the node does not exist, it is generated right before the next call.

Algorithm 3 *IncreaseSimilarity*(Document d_i, Document d_j, Word *word*, Integer *depth*)

```
1: if depth scoring is used then
2:     $improvement \leftarrow word.idfValue \times depth^{depthscorepower}$
3: else
4:     $improvement \leftarrow word.idfValue$
5: end if
6: $threshold_i \leftarrow ETL \times scs[d_i]$
7: $threshold_j \leftarrow ETL \times scs[d_j]$
8: $cs[d_i][d_j] \leftarrow cs[d_i][d_j] + improvement$
9: $cs[d_j][d_i] \leftarrow cs[d_j][d_i] + improvement$
10: if $cs[d_i][d_j] > threshold_i$ then
11:     $Containers_{d_i} \leftarrow Containers_{d_i} \cup d_j$
12: end if
13: if $cs[d_j][d_i] > threshold_j$ then
14:     $Containers_{d_j} \leftarrow Containers_{d_j} \cup d_i$
15: end if
```

3.3 Complexity Analysis

In this section, we discuss the performance of CoDet in two different extreme cases. For each scenario, let N denote the number of documents and let c denote the average number of words per document, like most of the previous works, it is considered as constant.

3.3.1 First Scenario: One Content, N Documents

In this case, each document has the same content; therefore, corpus tree contains exactly c nodes. Each node contains N integers in its visitor list. As a result, the memory requirement of the corpus tree is $O(N)$ but due to pairwise containment similarity increase operations, the algorithm takes $O(N^2)$ time and space since calculated similarity values are also saved. It is actually the worst case for CoDet. In order to decrease the running time and memory requirement, when two documents are labelled as near-duplicate one of them can be removed from the corpus tree since these documents would be contained by the same documents and containers can be found by only holding one of them in the corpus tree.

3.3.2 Second Scenario: N Different Contents, N Documents

In this case, each document has totally different content. Thus, corpus tree contains Nc nodes (one node for each term). Each node contains only one document id in its visitor list since each node is visited by exactly one document. Hence, asymptotically the memory requirement of the corpus tree is $O(N)$ and the algorithm takes $O(N)$ time.
The first scenario is the worst case for CoDet, where the algorithm performs non-linearly. The second one is the best case for CoDet and the algorithm runs in linear time. In practice the algorithm behaves as if it is linear because average number of near-duplicate per document is significantly smaller than N. For example, the number of near-duplicates per page follows a power-law distribution [14]. Also CoDet is especially suitable for streaming news since with a time window concept, which removes old documents from the corpus tree, it does not grow too much.

Chapter 4

Experimental Setup

For hashing purposes, the SHA1 [9] algorithm, which produces a 160-bit message digest, is used in all methods. As seen in Table 4.1, even a small change in the input (in the example, it is just one character) results in a substantially different hash value. In order to make a fair evaluation, parameters of each algorithm are optimized to give the best results for efficiency. We performed the experiments on a machine with quad 2.1Ghz six-core AMD Opteron processors with six 128 KB L1, 512 KB L2, and one 6MB L3 cache. It has 128 GB memory and operating system Debian Linux v5.0.5.

4.1 Algorithms in Comparison

We used four algorithms to compare their effectiveness and efficiency with CoDet. These algorithms are *DSC, Full Fingerprinting, I-Match, SimHash.* Number of

Table 4.1: SHA-1 Hash Function

Input	Hash Value
The quick brown fox jumps over the lazy dog	2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
The quick brown fox jumps over the lazy cog	de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3

used words to create fingerprints and shingles in the algorithms are optimized empirically. Our dataset consists of news documents so they are generally small sized. Therefore, the optimal values of the parameters are also small.

4.1.1 DSC

DSC [4] is a shingling based method for detecting near-duplicates. Its working procedure is as follows. Each document is tokenized into words and each *three* overlapping substrings of *four* consecutive words are hashed to create fingerprints. Each of these fingerprints is called as *shingle*. Let $S(d_A)$ and $S(d_C)$ represent all the shingles of document d_A and d_C respectively. To measure the similarity between d_A and d_C, Jaccard similarity between $S(d_A)$ and $S(d_C)$ is used. Broder called this as resemblance of documents which is shown by $r(d_A, d_C) = \frac{S(d_A) \cap S(d_C)}{S(d_A) \cup S(d_C)}$.

In our experiments, if a document d_C contains 60% of d_A's shingles, then, $d_C \supseteq d_A$.

4.1.2 Full Fingerprinting (FFP)

For each document, all substrings of size *four* are hashed. If document d_C contains 60% of d_A's hash values, then $d_C \supseteq d_A$.

The main difference between FFP and DSC is that FFP creates shingles for every substring of size *four* in a document; however, DSC creates shingles for *only three overlapping* substring of size *four*. Because of this, DSC requires less processing time than FFP, whose runtime is O(N^2) where N is the number of documents. As expected, Effectiveness of DSC is worse than FFP.

4.1.3 I-Match

I-Match is designed to detect duplicates in a large scale of document collections. The algorithm does not rely on simple parsing, it also uses collection statistics to identify which terms are worth to be used in comparison. ignores first *two* words with the highest *idf* values. After that, *ten* words with the highest *idf* values are used to create a fingerprint for each document. When a pair of documents has the same fingerprint, the pair is marked as containment.[9] First two words with the highest *idf* values are ignored since they are likely to be misspellings. Words with low idf values are not taken into account since they are usually common words in language and they do not represent the document at all. The running time of I-Match is O($NlogN$) in the worst case, where N is the number of documents and each document has the same content. In general, it works in linear time.

4.1.4 SimHash

SimHash is another fingerprinting based approach. Unlike to classical hash functions, in which a small change in the input results in a totally different output, SimHash is designed to give similar outputs for similar inputs. First *two* words with the highest *idf* values are ignored. Then, each unique term of a document is hashed. We use a vector v, whose size is equal to the hash value bit count, to determine the final SimHash [8] values. For each term t, the i^{th} element of vector v is updated as follows: If the i^{th} bit of the hash value of t is *zero*, then it is decreased by the *idf* of w. It is increased by the *idf* otherwise. The calculation of the SimHash values is shown in Table 4.2. Finally, if the i^{th} element of v is positive, the i_{th} bit of the SimHash value is set to *one*; otherwise it is set to *zero*. When a pair of documents' SimHash values has a *Hamming distance* less than *three*, then the pair is considered as *containment*. Construction of the SimHash values is done in linear time; however, calculation of bit differences is time consuming operation.

Table 4.2: SimHash Value Calculation for document d: "NASDAQ starts day with an increase.". For the sake of clarity, hash values are 4 bits long.

Word	**Hash**	**Weight**	1st **Bit**	2nd **Bit**	3rd **Bit**	4th **Bit**
Nasdaq	1100	0.10	0.10	0.10	-0.10	-0.10
starts	1011	0.04	0.04	-0.04	0.04	0.04
day	0100	0.03	-0.03	0.03	-0.03	-0.03
with	1111	0.02	0.02	0.02	0.02	0.02
an	1000	0.01	0.01	-0.01	-0.01	-0.01
increase	0110	0.06	-0.06	0.06	0.06	-0.06
Sum			0.08	0.16	-0.02	-0.14
SimHash Value			1	1	0	0

4.2 Test Collection Preparation

There is no gold-standard test collection for containment detection in news corpora; therefore, we prepared a test dataset from the Turkish TDT (Topic Detection and Tracking) news collection *(BilCol-2005)* [6] which contains 209,305 streaming (time-ordered) news articles obtained from five different Turkish web news sources. The statistics about the whole collection is provided in Table 7

For efficiency measurement, we used all documents of *BilCol-2005*. For effectiveness measurement, we used the first 5,000 documents. It is practically impossible to provide human assessment for each document pair in this sub-collection. Our approach to human assessments is similar to the pooling method used in TREC for the evaluation of IR systems [24]. For the creation of the dataset, we obtained a number of possible containments by running all five methods (including CoDet) with permissive parameters. In this way, methods nominate all pairs that would normally be chosen with their selective parameters, together with several additional pairs as containment candidates. Since the methods are executed with permissive parameters, we expect that most of the real containments will be added to the test collection. All pairs of documents, which are marked as containments by any of the methods, are brought to the attention of human assessors to determine whether they actually are containments. Note that in order to measure the effectiveness of a new algorithm with this test dataset, adding human assessments only for containment candidates that are nominated

Table 4.3: Information about distribution of stories among news sources in BilCol2005 (Can, F. Kocberler, S., Balcik, E., Kaynak, C., Ocalan, H. C., Uyar, E. “New event detection and topic tracking in Turkish”, JASIST, Vol. 61:4, ©2010 ASIS&T, http://dx.doi.org/10.1002/asi.21264. Reprinted by permission.)

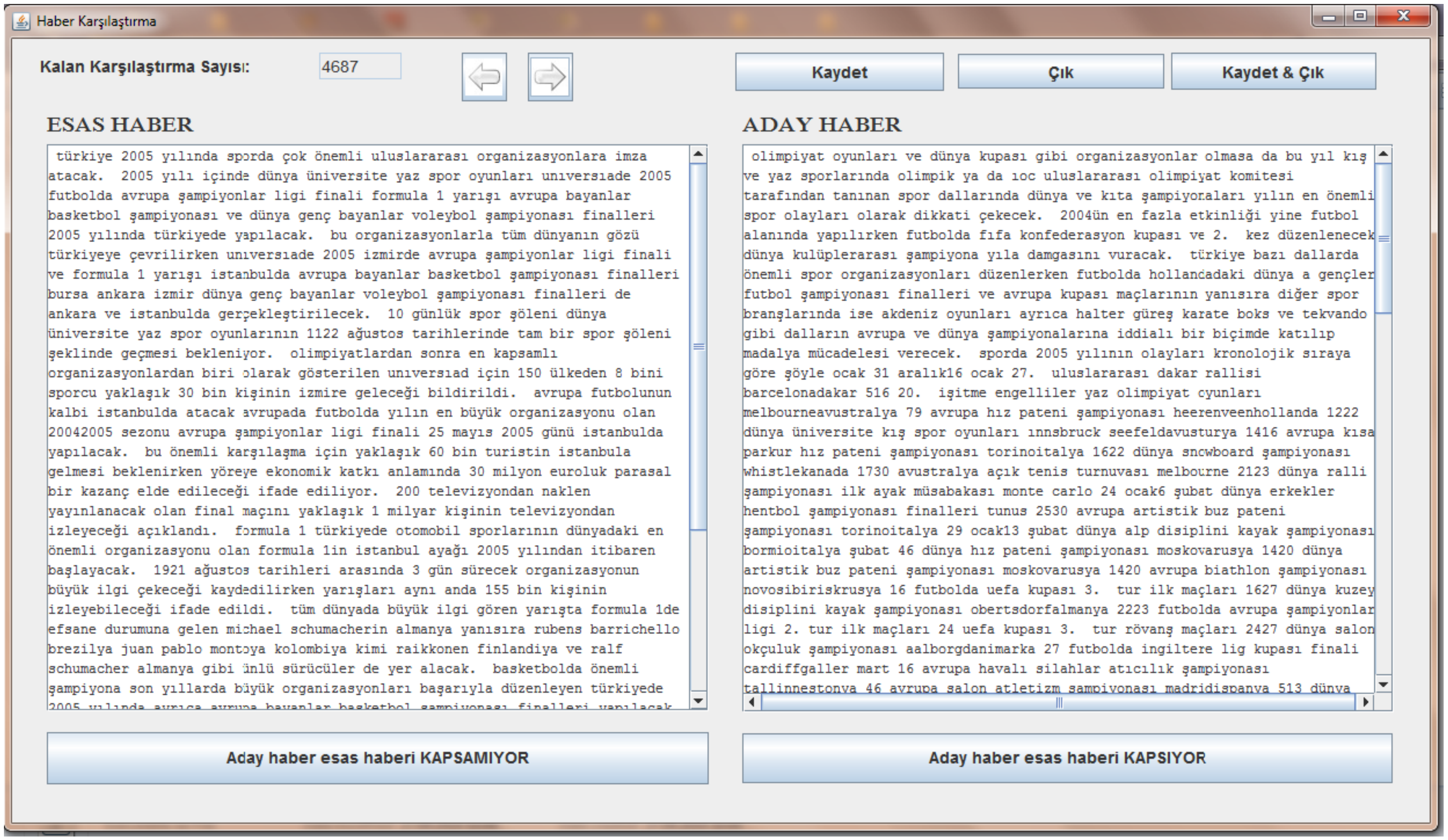
News Source	**No. of News Stories**	**Percent of All Stories**	**Download Amount (MB)**	**Net Amount (MB)**	**Avg. No. of Words per Story**
CNN Turk	23,644	11.3	1,008.3	66.8	271
Haber 7	51,908	24.8	3,629.5	107.9	238
Milliyet Gazetesi	72,233	24.8	3,629.5	107.9	238
TRT	72,233	34.5	508.3	122.5	218
Zaman Gazetesi	42,530	20.3	45.3	33.7	97
All together	209,305	100.0	6,129.3	349.2	196*

* Different from the weighted sum of the average word lengths due to rounding error.

solely by this new algorithm to our dataset is sufficient.

By this approach, our dataset includes only true positive *(TP)* and false positive *(FP)* document pairs returned by any of our permissive algorithms. Exclusion of true negative and false negative pairs do not change the relative effectiveness rankings of selective algorithms during the test phase; because, if a permissive algorithm marks a pair as negative (non-containment), then its selective counterpart should also marks that pair as negative. Therefore, including TN and FN pairs of permissive algorithms in our dataset would not contribute to the number of positive pairs (TPs and FPs) returned by any selective algorithm during the test phase. Hence, using our pruned dataset, precision[1] values of the selective algorithms remain unchanged with respect to precision values they would obtain in a full dataset having annotations for all possible document pairs. Similarly, recall[2] values of the selective algorithms decrease proportionally (with the same ratio of total number of containments in the pruned dataset to the total number of containments in the full dataset, for all algorithms) with respect to recall values they would obtain in the full dataset.

[1]Precision $(P) = |TP|/(|TP| + |FP|)$
[2]Recall $(R) = |TP|/(|TP| + |FN|)$


Haber Karşılaştırma
Kalan Karşılaştırma Sayısı:
4687
Kaydet
Çık
Kaydet & Çık
ESAS HABER
ADAY HABER
Aday haber esas haberi KAPSAMIYOR
Aday haber esas haberi KAPSIYOR

```
 türkiye 2005 yılında sporda çok önemli uluslararası organizasyonlara imza
atacak.  2005 yılı içinde dünya üniversite yaz spor oyunları universiade 2005
futbolda avrupa şampiyonlar ligi finali formula 1 yarışı avrupa bayanlar
basketbol şampiyonası ve dünya genç bayanlar voleybol şampiyonası finalleri
2005 yılında türkiyede yapılacak.  bu organizasyonlarla tüm dünyanın gözü
türkiyeye çevrilirken universiade 2005 izmirde avrupa şampiyonlar ligi finali
ve formula 1 yarışı istanbulda avrupa bayanlar basketbol şampiyonası finalleri
bursa ankara izmir dünya genç bayanlar voleybol şampiyonası finalleri de
ankara ve istanbulda gerçekleştirilecek.  10 günlük spor şöleni dünya
üniversite yaz spor oyunlarının 1122 ağustos tarihlerinde tam bir spor şöleni
şeklinde geçmesi bekleniyor.  olimpiyatlardan sonra en kapsamlı
organizasyonlardan biri olarak gösterilen universiad için 150 ülkeden 8 bini
sporcu yaklaşık 30 bin kişinin izmire geleceği bildirildi.  avrupa futbolunun
kalbi istanbulda atacak avrupada futbolda yılın en büyük organizasyonu olan
20042005 sezonu avrupa şampiyonlar ligi finali 25 mayıs 2005 günü istanbulda
yapılacak.  bu önemli karşılaşma için yaklaşık 60 bin turistin istanbula
gelmesi beklenirken yöreye ekonomik katkı anlamında 30 milyon euroluk parasal
bir kazanç elde edileceği ifade ediliyor.  200 televizyondan naklen
yayınlanacak olan final maçını yaklaşık 1 milyar kişinin televizyondan
izleyeceği açıklandı.  formula 1 türkiyede otomobil sporlarının dünyadaki en
önemli organizasyonu olan formula 1in istanbul ayağı 2005 yılından itibaren
başlayacak.  1921 ağustos tarihleri arasında 3 gün sürecek organizasyonun
büyük ilgi çekeceği kaydedilirken yarışları aynı anda 155 bin kişinin
izleyebileceği ifade edildi.  tüm dünyada büyük ilgi gören yarışta formula 1de
efsane durumuna gelen michael schumacherin almanya yanısıra rubens barrichello
brezilya juan pablo montoya kolombiya kimi raikkonen finlandiya ve ralf
schumacher almanya gibi ünlü sürücüler de yer alacak.  basketbolda önemli
şampiyona son yıllarda büyük organizasyonları başarıyla düzenleyen türkiyede
2005 yılında ayrıca avrupa bayanlar basketbol şampiyonası finalleri yapılacak
```

```
 olimpiyat oyunları ve dünya kupası gibi organizasyonlar olmasa da bu yıl kış
ve yaz sporlarında olimpik ya da ıoc uluslararası olimpiyat komitesi
tarafından tanınan spor dallarında dünya ve kıta şampiyonaları yılın en önemli
spor olayları olarak dikkati çekecek.  2004ün en fazla etkinliği yine futbol
alanında yapılırken futbolda fifa konfederasyon kupası ve 2.  kez düzenlenecek
dünya kulüplerarası şampiyona yıla damgasını vuracak.  türkiye bazı dallarda
önemli spor organizasyonları düzenlerken futbolda hollandadaki dünya a gençler
futbol şampiyonası finalleri ve avrupa kupası maçlarının yanısıra diğer spor
branşlarında ise akdeniz oyunları ayrıca halter güreş karate boks ve tekvando
gibi dalların avrupa ve dünya şampiyonalarına iddialı bir biçimde katılıp
madalya mücadelesi verecek.  sporda 2005 yılının olayları kronolojik sıraya
göre şöyle ocak 31 aralık16 ocak 27.  uluslararası dakar rallisi
barcelonadakar 516 20.  işitme engelliler yaz olimpiyat oyunları
melbourneavustralya 79 avrupa hız pateni şampiyonası heerenveenhollanda 1222
dünya üniversite kış spor oyunları ınnsbruck seefeldavusturya 1416 avrupa kısa
parkur hız pateni şampiyonası torinoitalya 1622 dünya snowboard şampiyonası
whistlekanada 1730 avustralya açık tenis turnuvası melbourne 2123 dünya ralli
şampiyonası ilk ayak müsabakası monte carlo 24 ocak6 şubat dünya erkekler
hentbol şampiyonası finalleri tunus 2530 avrupa artistik buz pateni
şampiyonası torinoitalya 29 ocak13 şubat dünya alp disiplini kayak şampiyonası
bormioitalya şubat 46 dünya hız pateni şampiyonası moskovarusya 1420 dünya
artistik buz pateni şampiyonası moskovarusya 1420 avrupa biathlon şampiyonası
novosibiriskrusya 16 futbolda uefa kupası 3.  tur ilk maçları 1627 dünya kuzey
disiplini kayak şampiyonası obertsdorfalmanya 2223 futbolda avrupa şampiyonlar
ligi 2. tur ilk maçları 24 uefa kupası 3.  tur rövanş maçları 2427 dünya salon
okçuluk şampiyonası aalborgdanimarka 27 futbolda ingiltere lig kupası finali
cardiffgaller mart 16 avrupa havalı silahlar atıcılık şampiyonası
tallinnestonya 46 avrupa salon atletizm şampiyonası madridispanya 513 dünya
```

Figure 4.1: Comparison Screen of News Containment Control

Our pooling process generated 4,727 document pairs nominations. We performed a human-based annotation to obtain a ground truth. The pooled document pairs are divided into 20 groups containing about the same number of nominations. Each document pair is annotated by two assessors. The assessors are asked if the nominated document pairs are actually containments. In order to make the comparisons, annotators use a java program called *News Containment Control*. In this program, users are first shown the definition of containment: *"if a reader that already read the candidate document, does not gain anything by reading the processing document, then, candidate document is said to contain processing document."*. Subsequently, they start to compare selected pairs by seeing both processing and candidate document in the same screen. Users are allowed to save their profile and quit to program and they can continue the evaluation later. They can also have an option to change their previous answers. A sample comparison screen can be seen in Fig. 4.1.

The assessors identified 2,875 containment cases. The size of our dataset is comparable with the annotated test collections reported in related studies [19].

In information retrieval, human assessors may have different opinions about the relevance of a document to a query. A similar situation arises in our assessments. For example, for the document pair d_C = "XYZ shares increase 10% from 100 to 110." and d_A = "XYZ shares increase from 100 to 110.", some assessors may say that d_C and d_A are near-duplicates, while some others may claim d_C contains d_A, but d_A does not contain d_C. In such cases, we expect disagreements among human assessors. In order to validate the reliability of the assessments, we measured the agreements of the judgements by using the *Cohen's Kappa measure*, and obtained an average agreement rate of 0.73. This indicates substantial agreement [16], which is an important evidence for the reliability of our test dataset. Furthermore, such conflicts are resolved by an additional assessor.

Chapter 5

Experimental Results

In this chapter, we discuss the effects of well known information retrieval techniques that are applied after document parsing and just before similarity measure. These techniques are *stemming, eliminating stop words*, and *idf value calculation*. Then, we investigate the impacts of the following parameters on CoDet's performance: Processed Suffix Count (PSC), Depth Threshold (DT), and Word Sorting (WS). This discussion is followed by efficiency and effectiveness performance of CoDet with those of four well-known near-duplicate detection algorithms. Effectiveness measurement is done by precision (P), recall (R) and F_1[3] score values. Effectiveness experiments are conducted on the prepared test collection. Efficiency experiment is performed with the whole *BilCol-2005*.

5.1 Impacts of IR Techniques

These tecnhniques are proven to have positive impact on near-duplicate detection. Therefore, we do not test them on algorithms in comparison. In the experiments of these other algorithms, these techniques are all applied. Nevertheless, we would like to observe their effects on CoDet's performance separately. Hence, we test both their enabled and disabled cases in CoDet runs.

[3] $F_1 = 2PR/(P + R)$

5.1.1 Stemming

In information retrieval, stemming is the process of reducing words to their stem, which is their root form. Can et al. [7] state that stemming has a significant effect on information retrieval in Turkish but CoDet and other algorithms used in this research do not make any semantic analysis; hence, we consider two cases: no stemming and first-5, 5-prefix characters of each word. No stemming case is not considered for the other algorithms since its effectiveness is already proven. We would like to observe the effect of stemming on CoDet's performance so both no stemming and first-5 methods are tested for CoDet. When all the other parameters are optimized, no stemming gives F_1 score of 0.74 while first-5 improves F_1 score to 0.76.

5.1.2 Eliminating stop words

Stop words are the words, which are filtered out prior to processing documents. In the experiments, we use our own stop word list by taking the most frequent words in our collection, which contains 180 words and can be seen in Table 2.1. As expected, elimination of stop words does not affect the effectiveness. Without elimination, again F_1 score of 0.76 is achieved since stop words have almost 0 *idf* value implying that they are in fact ignored by CoDet even if they are not eliminated. Interestingly, elimination of stop words does not influence efficiency considerably, as it decreases corpus tree size by 1% only. In our opinion, this is caused by *depththresholding* and *wordsorting*. Since we sort words of a sentence with respect to their *idf values* and we only use the first six words, stop words are not processed at all because it is difficult for them to be in the first six.

5.1.3 IDF Value Calculation

Inverted document frequency (idf) measures to what extent general importance of a word in a corpus. As can be seen in Equation 2.1 it is obtained for a word by dividing the total number of documents by number of documents that contain the word, then quotient's logarithm is taken. In the simplest form, term frequency (tf) is taken as term appearance count divided by total term count for a document (Equation 2.2). In the experiments, it is found that *idf* gives better results than *tf* and *tf-idf*. CoDet does not make use of *tf values* since container documents may be substantially longer than contained ones; thus, they have higher term counts and this changes *tf values*. Due to the structure of our similarity measure, we want each word t to have the same impact value for documents containing w.

Idf values are attained on last 25% of our dataset. By this way, we make sure that effectiveness experiments that are conducted on first 5000 documents do not gain any unfair advantage.

5.2 Impacts of Parameters

5.2.1 Processed Suffix Count (PSC)

PSC determines how many suffixes of each sentence are inserted to the corpus tree. If the PSC is 3, the processed suffixes for "NASDAQ starts day with an increase." are the sentence itself, $< starts, day, with, an, increase >$ and $< day, with, an, increase >$. Raising PSC increases space requirement but does not change the effectiveness as shown in Fig. 5.1. Different PSC values result in close F_1 scores.

5.2.2 Depth Threshold (DT)

DT determines how many words of a sentence are processed. If DT is 3, then the processed words for "NASDAQ starts day with an increase." are $< nasdaq, starts, day >$. Fig. 5.2 and 5.3 show the effect of DT on F_1 score. Sorting words of a sentence by *idf* values places representative words close to the *virtual root*. Thus, results are better for small DT values when words sorting is enabled. It avoids the noise effect of insignificant words in similarity calculations.

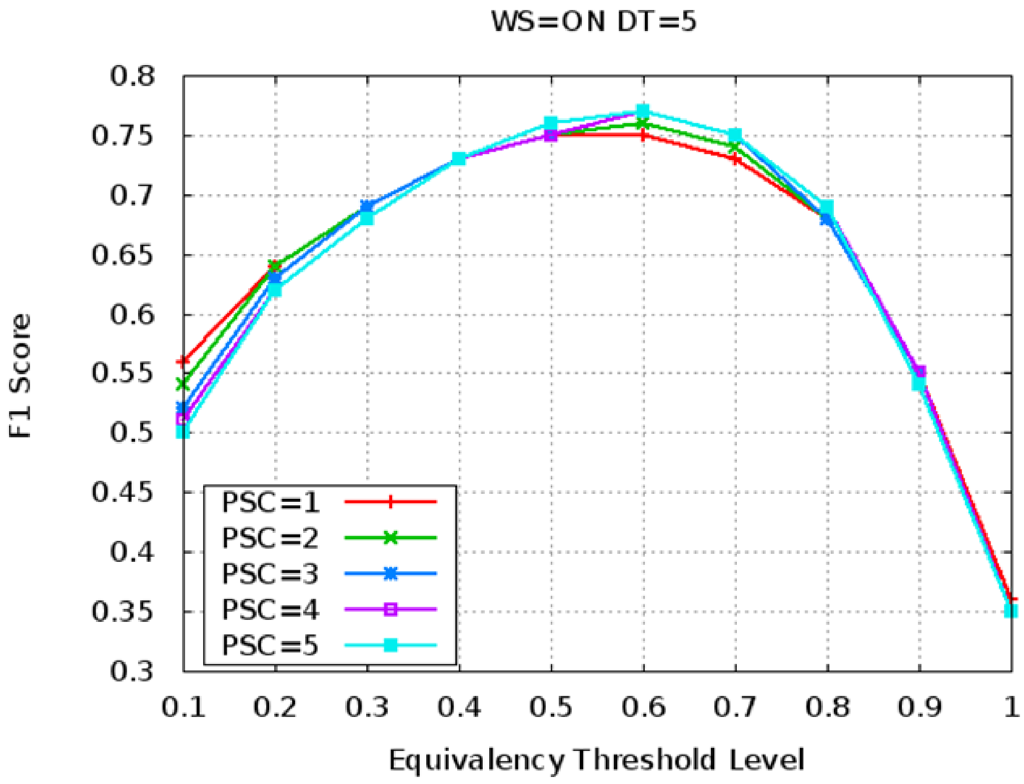


Figure 5.1: Effect of Processed Suffix Count (PSC) (Varol, E. Can, F., Aykanat, C., Kaya, O. "CoDet: Sentence-based Containment Detection in News Corpora", ACM CIKM '11 Conf., ©2011 ACM, Inc. http://dx.doi.org/10.1145/2063576.2063887. Reprinted by permission.)

In the experiments, DT value of 5 gives the best result; also smaller DT values yield a similar performance. Hence, instead of having the corpus tree structure, an algorithm that considers only a few most significant words from each sentence can improve efficiency without sacrificing effectiveness significantly.

5.2.3 Word Sorting (WS)

Sorting words in sentences by *idf* values causes important words to be located close to the virtual root. Since most sentences start with common words, by using word sorting, we avoid many redundant similarity calculations. In the

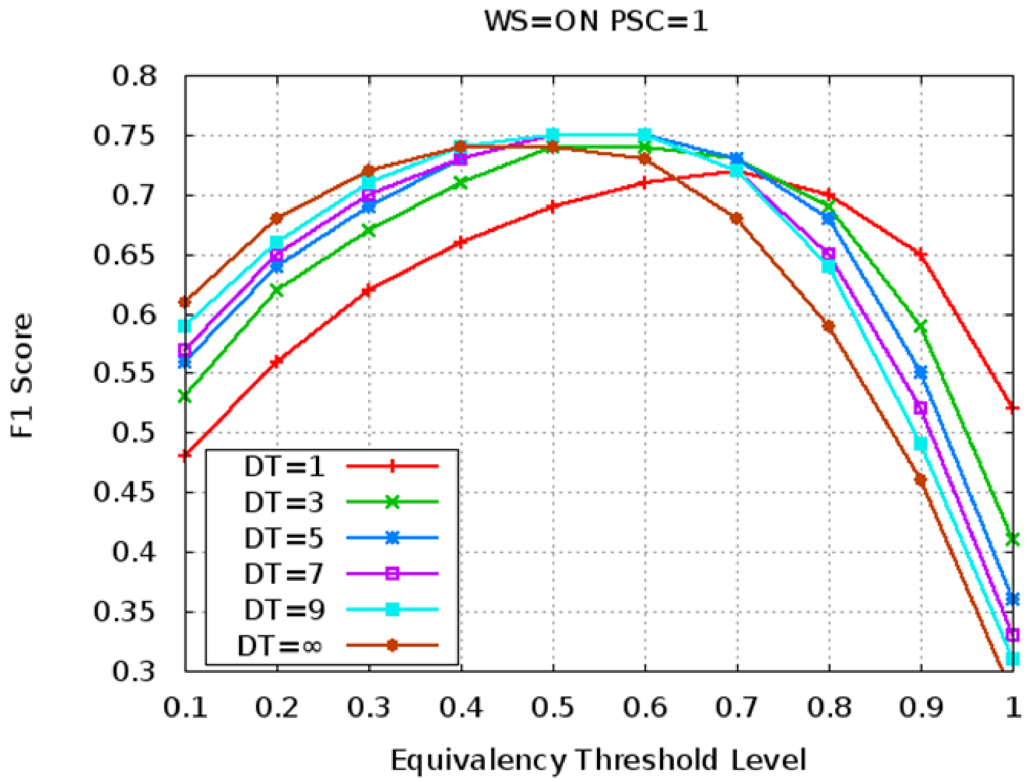


Figure 5.2: Effect of Depth Threshold (DT): Word Sorting (WS) is on (Varol, E. Can, F., Aykanat, C., Kaya, O. "CoDet: Sentence-based Containment Detection in News Corpora", ACM CIKM '11 Conf., ©2011 ACM, Inc. http://dx.doi.org/10.1145/2063576.2063887. Reprinted by permission.)

experiments, enabling words sorting decreases average number of calculated similarity values per document from 341 to 3.53. Additionally, enabling word sorting decreases the noise effect of the common words and produces better results as seen in Fig. 5.2 and Fig. 5.3.

5.2.4 Depth Score Power (DSP)

Idf value is not the only factor used in calculating impact values of nodes on similarity. The contribution of depth on the impact value is also tested. The motivation behind is that if depth of a node n is high, it means that documents in its visitor list are already contained by visitor list of nodes on the path from

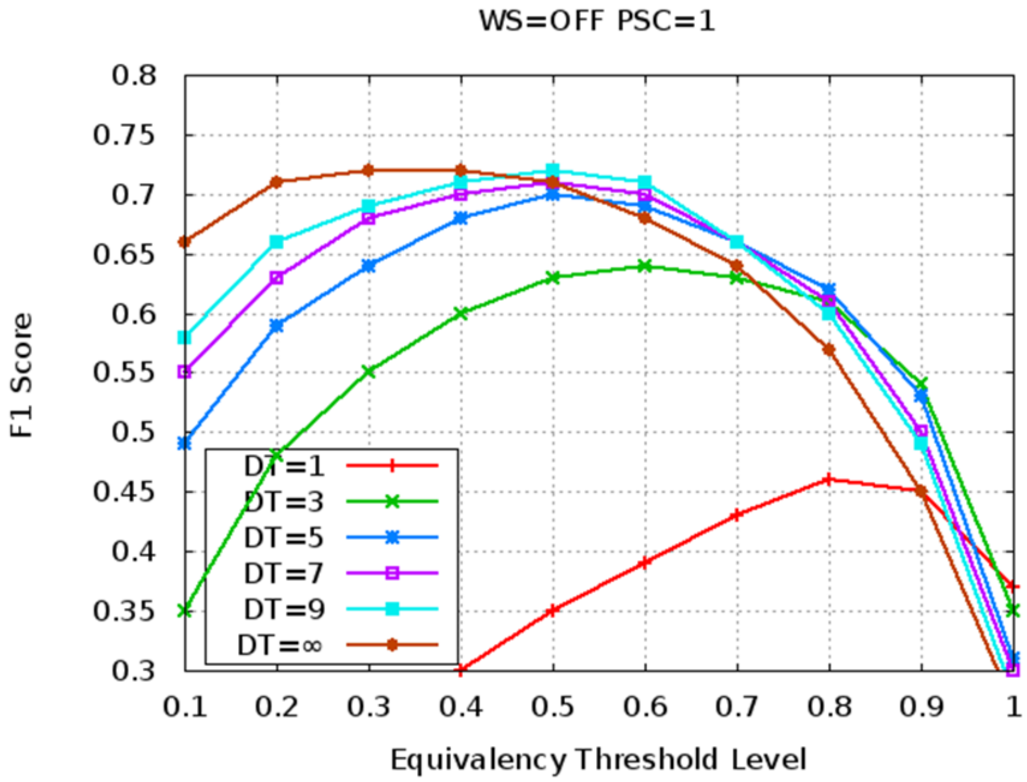


Figure 5.3: Effect of Depth Threshold (DT): Word Sorting (WS) is off (Varol, E. Can, F., Aykanat, C., Kaya, O. "CoDet: Sentence-based Containment Detection in News Corpora", ACM CIKM '11 Conf., ©2011 ACM, Inc. http://dx.doi.org/10.1145/2063576.2063887. Reprinted by permission.)

virtualrootnode to *n*. Additionally, if word sorting is enabled, terms with low *idf* values do not have any considerable impact even if they are in common between two sentences. Depth scoring would give them an opportunity to affect if they are in the bottom part of the corpus tree. In the experiments, only values between 0 and 3 are tested for *DSP* since we do not see any room for improvement after 3. Taking *DSP* as zero means that no effect of depth and results are worse than the others. There is no much difference between 1 and 2. When *DSP* is 3, the depth factor starts to show a sign of domination and causes to a degradation in quality. Detailed results can be found in Table 5.1

Table 5.1: Effect of Depth Score Power (DSP)

DSP	**Precision**	**Recall**	F_1 **Measure**
0	0.60	0.62	0.61
1	0.75	0.76	0.76
2	0.76	0.75	0.76
3	0.55	0.53	0.54

Table 5.2: Effectiveness Comparison

Algorithm	**Precision**	**Recall**	F_1 **Measure**
FFP	0.82	0.88	0.85
CoDet	0.75	0.76	0.76
I-Match	0.72	0.33	0.45
SimHash	0.53	0.30	0.39
DSC	0.22	0.45	0.30

5.3 Comparing with the Other Algorithms

The efficiency results are given in Fig. 5.4. As the number of documents increase, execution time of full fingerprinting increases non-linearly. It calculates similarity values for each document pair that has at least one substring in common. Hence, its running time is $O(N^2)$ where N is the number of documents. CoDet performs as the third best algorithm in time efficiency since the corpus tree accesses impose many random memory accesses, which disturb cache coherency. Additionally, if two documents visits at least one node in common; then, a similarity value is calculated for them. Therefore, average number of containment count per document is an important statistic for the performance of CoDet. Our results show that I-Match, SimHash and CoDet are scalable to large collections.

Table 5.2 shows the effectiveness results. The best performance with a value of 0.85 F_1 score is observed with FFP since it calculates text overlaps between document pairs having a common substring. Therefore, without making any semantic analysis, it is difficult to outperform FFP in terms of effectiveness with a time-linear algorithm. CoDet finds text overlaps by only using important words of sentences and is the second best in terms of effectiveness with an F_1 score of

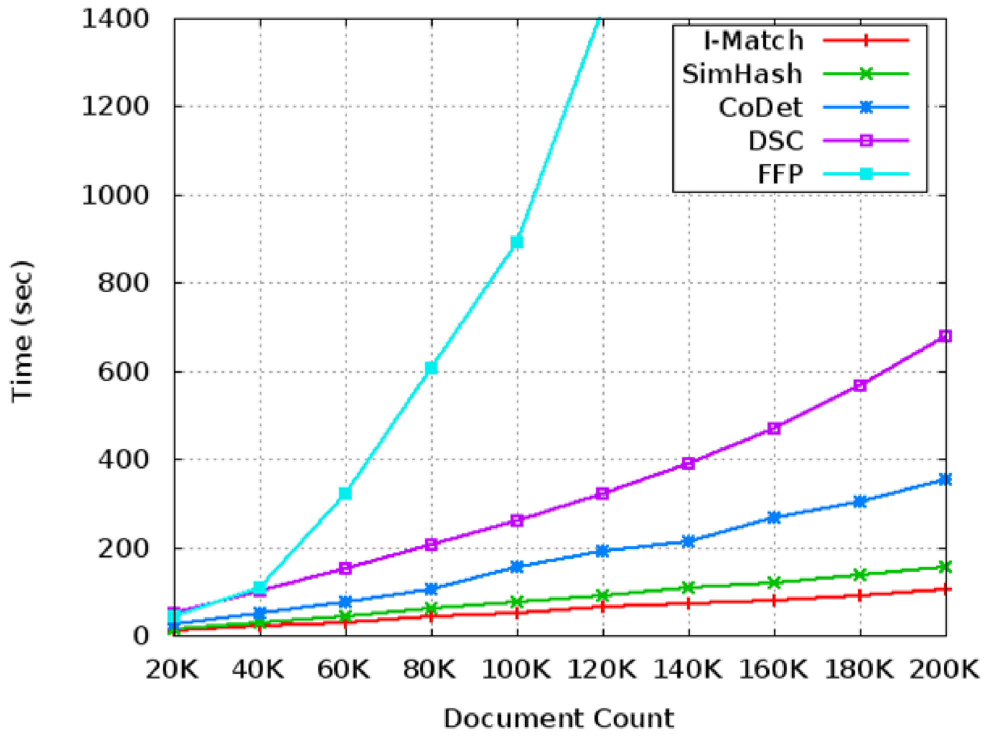


Figure 5.4: Efficiency Comparison: Execution Time vs. Document Count (Varol, E. Can, F., Aykanat, C., Kaya, O. "CoDet: Sentence-based Containment Detection in News Corpora", ACM CIKM '11 Conf., ©2011 ACM, Inc. http://dx.doi.org/10.1145/2063576.2063887. Reprinted by permission.)

0.76. I-Match, SimHash, and DSC perform poorly with respective F_1 scores of 0.45, 0.39, and 0.30. FFP is not a feasible choice for large collections due to its time complexity; thus, CoDet is a preferable algorithm for containment detection in most text corpora.

Chapter 6

Future Work

There are several future research directions for our work. Some possibilities are listed in the following.

- ◇ In news reporting, the most important information is generally placed at the beginning of the text. For improving the effectiveness of CoDet we may use a term weighting approach which assigns higher weights to terms which are closer to the beginning of the document.
- ◇ In our implementation of CoDet we did not utilize metadata; e.g. news category or news title. Utilizing metadata can improve effectiveness. However, it requires a new dataset since our existing dataset does not provide such metadata.
- ◇ In the offline usage of CoDet for a general corpus, we can lexicographically sort the sentences of the documents. By this way we can process similar sentences simultaneously. This can substantially improve the efficiency due to the locality of memory references during insertions to the corpus tree, since we expect to traverse the same nodes in the tree for similar sentences.
- ◇ CoDet can be combined with other methods having complementary features to improve its effectiveness by maintaining a practical efficiency.

◇ In the experiments, we used *idf* value depth of the processing node for term weighting. Rather than this, we only check the effect of *tf* value. Different term weighting approach may improve the result quality.

◇ CoDet can be used in the web environment. In such collections it is highly parallelizable after dividing documents into sub-categories. After categorization, we expect containment pairs to appear in the same group; therefore, each category can be processed independently. Additionally, without categorization, it can still be easily parallelized by letting each different machine to store a part of the corpus tree. As alphabetical partition can be used for this purpose, more balanced partitions can yield better performance.

◇ We would like to integrate our new containment detection algorihtm to Bilkent News Portal [5]. When multiple news sources is used for a system, near-duplicate detection has to be done since almost the same content arrives more than once from different news sources that are fed from the same news agencies.

◇ There may be some noise in especially auto-crawled text documents. For example, sometimes news documents have advertisements in our collection. The same advertisement can be found in many documents. Therefore, if two documents share only one or two complete sentences, these are most likely to be advertisements. CoDet is a sentence-based algorithm so it can be used for spam and advertisement filtering.

◇ Containment detection can be employed for plagiarism detection with some minor modifications. We already finish implementation and conduct some experiments and it is seen that the initial results are promising.

Chapter 7

Conclusions

Containment detection is an important problem which has its potential uses in many different areas such as preventing redundant information to news readers and detecting plagiarism. In this work we investigate this problem, which is a more generalized version of the near-duplicate detection problem. We introduce a new approach named CoDet. It simply processes documents sentence by sentence and inserts each sentence into the corpus tree word by word. In the corpus tree, each node contains a list of previously visitor documents. When a new document reaches to a node, its similarity values with previously visitor documents are increased. When similarity value of a document pair exceeds a threshold depending on the content of the document, the situation marked as containment. We also compare CoDet's performance with four other well-known methods(DSC, full fingerprinting, I-Match, and SimHash). In order to make a fair evaluation, these algorithms are adapted to containment detection. The corpus we worked on (*BilCol-2005*) consisted of 209,305 streaming (time-ordered) news articles obtained from five different Turkish web news sources. For efficiency measurement, we used all documents of *BilCol-2005.* For effectiveness measurement, we used the first 5,000 documents of it. With the help of pooling approach and human assessment, we construct a ground truth for this small corpus.

As the experimental results demonstrate CoDet is preferable to all these methods; since it produces considerably better results in a feasible time. It also has desirable features such as time-linear efficiency and scalability, which enriches its practical value. Our method is versatile, can be improved, and can be extended to different problem domains such as plagiarism detection.

Bibliography

[1] E. Airio. Word normalization and decompounding in mono-and bilingual ir. *Information Retrieval*, 9(3):249–271, 2006.

[2] S. Brin, J. Davis, and H. Garcia-Molina. Copy detection mechanisms for digital documents. In *ACM SIGMOD Conf.*, pages 398–409, 1995.

[3] A. Broder. On the resemblance and containment of documents. In *Proc. of Compression and Complexity of Sequences*, page 21, 1997.

[4] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8–13):1157–1166, 1997.

[5] F. Can, S. Kocberber, O. Baglioglu, S. Kardas, H. Ocalan, and E. Uyar. Bilkent news portal: a personalizable system with new event detection and tracking capabilities. In *ACM SIGIR Conf*, page 885, 2008.

[6] F. Can, S. Kocberber, O. Baglioglu, S. Kardas, H. C. Ocalan, and E. Uyar. New event detection and topic tracking in Turkish. *JASIST*, 61(4):802–819, 2010.

[7] F. Can, S. Kocberber, E. Balcik, C. Kaynak, C. Ocalan, and O. M. Vursavas. Information retrieval on Turkish texts. *JASIST*, 59(2):407–421, 2008.

[8] M. Charikar. Similarity estimation techniques from rounding algorithms. In *ACM STOC*, pages 380–388, 2002.

[9] A. Chowdury, O. Frieder, D. Grossman, and M. C. McCabe. Collection statistics for fast duplicate document detection. *ACM TOIS*, 20(2):171–191, 2002.

[10] J. G. Conrad and C. P. Schriber. Managing déjà vu: Collection building for the identification of duplicate documents. *JASIST*, 57(7):921–923, 2006.

[11] F. Deng and D. Rafiei. Approximately detecting duplicates for streaming data using stable bloom filters. In *ACM SIGMOD Conf.*, pages 25–36, 2006.

[12] L. Dolamic and J. Savoy. Stemming approaches for east european languages. *Advances in Multilingual and Multimodal Information Retrieval*, pages 37–44, 2008.

[13] H. Hajishirzi, W. Yih, and A. Kolcz. Adaptive near-duplicate detection via similarity learning. In *ACM SIGIR Conf.*, pages 419–426, 2010.

[14] M. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *ACM SIGIR Conf.*, pages 284–291, 2006.

[15] B. Jongejan and H. Dalianis. Automatic training of lemmatization rules that handle morphological changes in pre-, in-and suffixes alike. In *ACL AFNLP Conf*, pages 145–153, 2009.

[16] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[17] G. S. Manku and A. D. Jain. Detecting near-duplicates for web crawling. In *ACM WWW Conf.*, pages 141–150, 2007.

[18] K. Monostori, A. B. Zaslavsky, and H. W. Schmidt. Efficiency of data structures for detecting overlaps in digital documents. In *ACSC*, pages 140–147, 2001.

[19] M. Theobald, J. Siddharth, and A. Paepcke. Spotsigs: Robust and efficient near duplicate detection in large web collections. In *ACM SIGIR Conf.*, pages 563–570, 2008.

[20] E. Varol, F. Can, C. Aykanat, and O. Kaya. Codet: Sentence-based containment detection in news corpora. In *ACM CIKM Conf.*, pages 2049–2052, 2011.

[21] J. H. Wang and H. C. Chang. Exploiting sentence-level features for near-duplicate document detection. In *AIRS Conf.*, pages 205–217, 2009.

[22] H. Yang and H. C. Chang. Near-duplicate detection by instance-level constrained clustering. In *ACM SIGIR Conf.*, pages 421–428, 2006.

[23] Q. Zhang, Y. Zhang, H. Yu, and X. Huang. Efficient partial-duplicate detection based on sequence matching. In *ACM SIGIR Conf.*, pages 675–682, 2010.

[24] J. Zobel. How reliable are the results of large-scale information retrieval experiments. In *ACM SIGIR Conf.*, pages 307–314, 1998.

[25] J. Zobel and Y. Bernstein. The case of the duplicate documents measurement, search, and science. *LNCS*, 3841:26–39, 2006.

List of Symbols

d, d_A, d_B, d_C	Text documents
w	A word
IDF(w)	Inverse document frequency is a general importance of a word in a collection
TF(w,d)	Term frequency of word w in a document d
r(d_A,d_C)	Resemblance of documents
P (Precision)	Fraction of retrieved instances that are relevant
R (Recall)	Fraction of relevant instances that are retrieved
F_1 (F_1 Score)	A measure of test accuracy considering both P and R
D	A document collection containing $\|D\|$ documents
d_i	A document from D
t_i	A term
κ	Cohen's Kappa, a statistical measure of inter-annotator agreement
P(a)	Probability of agreement
P(c)	Probability of agreement by chance
A, B	Annotators
s_i, s_j	Sentences
S_A, S_B	Sets of sentences
CS(d_C,d_A)	Containment similarity function on documents
SCS(d_A)	Self containment similarity function on documents
cs(s_i,s_j)	Containment similarity function on sentences
f(s_i,s_j)	Longest word prefix match of sentences
$len_{f(s_i,s_j)}$	Length of longest word prefix match of sentences
$w_{f(s_i,s_j),k}$	k^{th} word of longest word prefix match of sentences