A NEW APPROACH TO SEARCH RESULT CLUSTERING AND LABELING

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

Anıl Türel

August, 2011

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Fazlı Can(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. İbrahim Körpeoğlu

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Seyit Koçberber

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of Graduate School of Engineering and Science

ABSTRACT

A NEW APPROACH TO SEARCH RESULT CLUSTERING AND LABELING

Anıl Türel
M.S. in Computer Engineering
Supervisor: Prof. Dr. Fazlı Can
August, 2011

Search engines present query results as a long ordered list of web snippets divided into several pages. Post-processing of information retrieval results for easier access to the desired information is an important research problem. A post-processing technique is clustering search results by topics and labeling these groups to reflect the topic of each cluster. In this thesis, we present a novel search result clustering approach to split the long list of documents returned by search engines into meaningfully grouped and labeled clusters. Our method emphasizes clustering quality by using cover coefficient and sequential k-means clustering algorithms. Cluster labeling is crucial because meaningless or confusing labels may mislead users to check wrong clusters for the query and lose extra time. Additionally, labels should reflect the contents of documents within the cluster accurately. To be able to label clusters effectively, a new cluster labeling method based on term weighting is introduced. We also present a new metric that employs precision and recall to assess the success of cluster labeling. We adopt a comparative evaluation strategy to derive the relative performance of the proposed method with respect to the two prominent search result clustering methods: Suffix Tree Clustering and Lingo. Moreover, we perform the experiments using the publicly available Ambient and ODP-239 datasets. Experimental results show that the proposed method can successfully achieve both clustering and labeling tasks.

Keywords: Search result clustering, cluster labeling, web information retrieval, clustering evaluation, labeling evaluation.

ÖZET

ARAMA SONUCU KÜMELEME VE ETİKETLEMEYE YENİ BİR YAKLAŞIM

Anıl Türel
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Fazlı Can
Ağustos, 2011

Arama motorları sorgu sonuçlarını sayfalara ayrılmış uzun web doküman listesi halinde sunmaktadır. Bilgi erişim sonuçlarının istenen bilgiye daha kolay ulaşmayı sağlamak amacıyla tekrar işlenmesi önemli bir araştırma konusudur. Bir tekrar işleme yöntemi de arama sonuçlarını konularına göre gruplamak ve bu grupları konularını yansıtacak şekilde etiketlemektir. Bu tezde, arama motorları tarafından oluşturulan uzun doküman listesini anlamlı bir şekilde gruplanmış ve etiketlenmiş kümelere ayıran yeni bir arama sonucu kümeleme yaklaşımı sunuyoruz. Metodumuz kapsama katsayısına dayalı kümeleme ve sıralı k-ortalamalar algoritmalarını kullanarak kümeleme kalitesine önem vermektedir. Diğer bir yandan, kümelerin etiketlemesi, anlamsız ya da kafa karıştıran etiketlerin kullanıcıları yanlış kümelere yönlendirerek zaman kaybettirmesi nedeniyle önemlidir. Bunlara ek olarak, bir kümenin etiketi, kümede bulunan dokümanların içeriklerini doğru bir biçimde yansıtmalıdır. Kümeleri etiketleme görevini etkin bir şekilde yerine getirebilmek için, terim ağırlıklandırmaya dayalı yeni bir küme etiketleme yöntemi sunulmaktadır. Ayrıca küme etiketlemenin başarısını değerlendirmek amacıyla hassasiyet ve kesinlik ölçütlerini kullanan yeni bir etiketleme metriği sunulmaktadır. Metodumuzun Sonek Ağacıyla Kümeleme ve Lingo gibi önde gelen arama sonucu kümeleme algoritmalarına göreceli performansını saptayabilmek amacıyla karşılaştırmalı bir değerlendirme yöntemi uygulanmaktadır. Diğer taraftan, herkesin kullanımına açık olan Ambient ve ODP-239 veri setlerinde testler gerçekleştirilmiştir. Test sonuçları önerilen metodun hem kümeleme hem de etiketleme görevini başarıyla yerine getirdiğini göstermektedir.

Anahtar sözcükler: Arama sonuçlarını kümeleme, küme etiketlemesi, web bilgi erişim, kümeleme değerlendirmesi, etiketleme değerlendirmesi.

Acknowledgement

First, I would like to thank my advisor, Prof. Dr. Fazlı Can, for his guidance, encouragement and support through my study.

This thesis would not have been possible without his contributions. I am very grateful to him for his patience during our research meetings and his endless confidence.

I also thank to the jury members, Assoc. Prof. Dr. Ibrahim Körpeoğlu and Assist. Prof. Dr. Seyit Koçberber for reading and reviewing my thesis.

I would like to thank to Bilkent Information Retrieval Group members: Ahmet Yeniçağ, Bilge Köroğlu, Cem Aksoy, Ceyhun Karbeyaz, Çağrı Toraman and Hayrettin Erdem for their support and friendship.

I am also appreciative of the financial support from the Scientific and Technical Research Council of Turkey (TÜBİTAK) under the grant number 108E074 and Bilkent University Computer Engineering Department.

Outside the academic life, there are some friends who directly or indirectly contributed to my completion of this thesis. I thank to Damla Arifoğlu, Dilek Demirbaş and İmren Altepe for their friendship, understanding and support.

Finally, I owe my loving thanks to my parents, Yakup and Fakriye Türel, my sister, Işıl Türel and my fiancé, İsmail Uyanık, for their undying love, support and encouragement.

to my family and my beloved fiancé

Contents

List of Figures

List of Tables

Chapter 1

Introduction

During information search, Internet users utilize web search engines. When a query is entered to a conventional search engine, relevant web sources are presented as a ranked list. Each result is shown as a snippet consisting of a title, URL and small text excerpt from the source website as shown in Figure 1.1. This figure demonstrates conventional presentation of search results as a ranked list returned for query "Turkey" by widely used search engine Google [1]. Clustering of search results is offered by Information Retrieval Community to enhance users search experience and decrease time needed for search. After then, search result clustering (SRC) task has been a popular research area of information retrieval [11, 43, 44, 46, 31].

Concisely, the SRC problem is based on dividing search results according to their subtopics and labeling these divisions to reflect the subtopic. Figure 1.2 demonstrates an open source search result clustering engine called Carrot^2 [42], which runs Lingo algorithm [31] in the background. It does not deal with indexing of web sources but gathers search results from different search engines and applies clustering and labeling onto these results before presenting to the user. On the left of the image, ten labels are provided that represent the clusters. Users look through cluster labels and select the one that is related to their information need. Then, search result clustering engine presents the results of that cluster. Users examine the snippets within that cluster. If users find the information they are

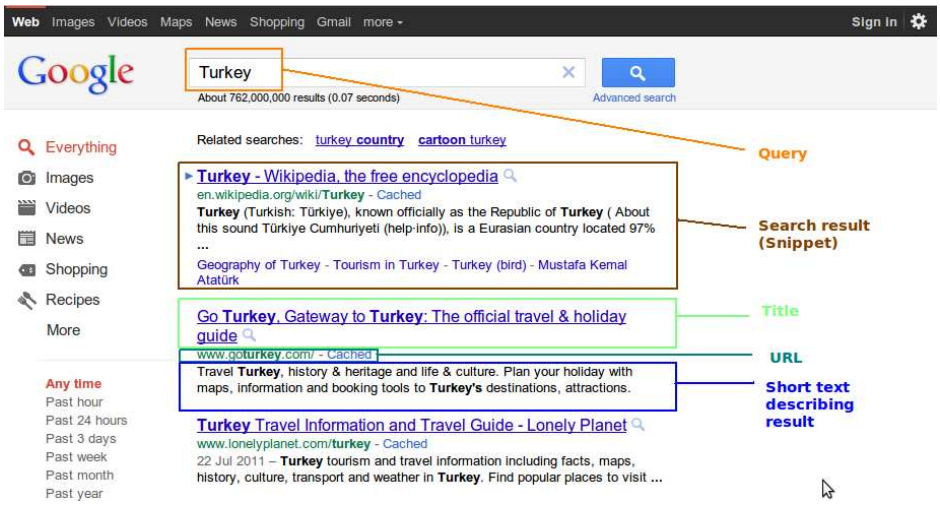

Figure 1.1: Screenshot is taken from widely used search engine Google where query "Turkey" is entered. Conventional search engine presents the search results as a list. The terminology used in this thesis is demonstrated. We use search result, result, document and snippet interchangeably in this thesis.

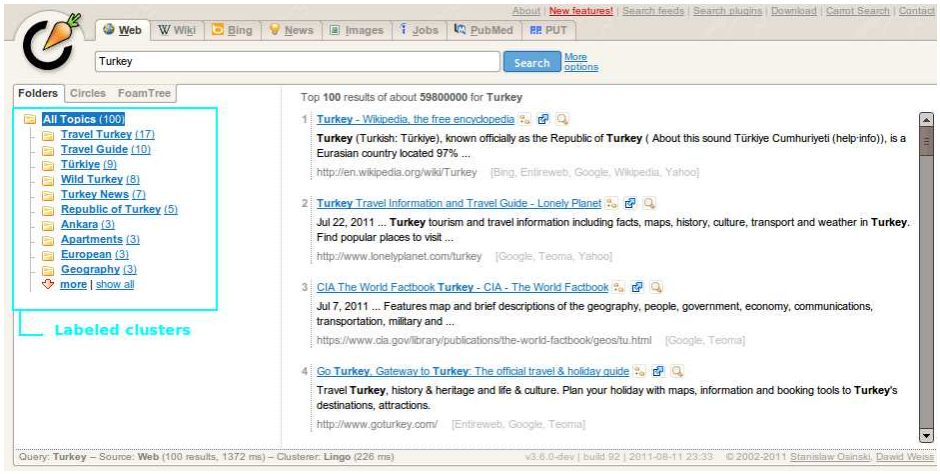

Figure 1.2: A search result clustering engine, Carrot^2 presents the cluster labels on the left for query "Turkey". Each label is followed by a number in parenthesis, for showing the number of snippets in that cluster.

looking for in a cluster, the number of search results to be reviewed diminishes in a logarithmic manner. Note that, conventional ranked list of all results are also reachable from the interface.

When a user enters a query to a clustering based search engine, query is firstly transmitted to a search engine. It employs its indexing structure and retrieves relevant results for the query. Then, it provides these results to the search result clustering system. Then, post-retrieval clustering mechanism starts to operate. SRC system filters the text data and extracts important features. It clusters and labels the input documents according to its algorithm and outputs labeled groups of results. Finally, clustered search results are presented from a web interface to be reviewed by users.

As the result of an entered query, SRC engine respond to users with all relevant search results and named groups of documents as shown in Figure 1.2. Users can also benefit from the ranked list with a conventional search behavior. They can also scan the labels of clusters. If one of them is connected to the subject of the question in their mind, they can click on that label. Clustering engine presents the documents in that cluster, and users go through these documents. They can change the cluster selection and explore the content of other clusters too. And hopefully, the clustered presentation helps the users during information exploration.

The goal of search result clustering task can be described as follows. An optimum search result clustering output is composed of thematic grouping related to the given query with meaningful and representative labels of the groups. Users read each label at a glance and naturally estimate the coverage of snippets inside that cluster. They decide whether results in each cluster is in accordance with the information need without looking inside the cluster. If they explore the cluster contents by clicking on the label, the snippets inside should satisfy the information need or at least increase the knowledge of users about the query.

1.1 Motivation

As information grows rapidly over the Internet, it gets harder for users to find the information they are looking for. After providing a query to a conventional search engine, a long list of search results divided into a lot of pages is presented. Similar results are scattered in the list, appearing in different pages. Without a proper arrangement of search results, finding the desired query result among ranked list of document snippets is usually difficult for most users. This problem is further aggravated when the query belongs to a general topic which contains documents from a variety of subtopics. At this point, the burden of solving interrelations among documents and extracting the relevant ones are left to the user.

In order to help web users during information exploration, post-processing of retrieval results are proposed to decrease time needed for search and enhance search experience of users. Search result clustering as being one of these enhancement attempts, presents labeled groups of search results in addition to the ranked list. Clusters contain documents about a subtopic of the query and each cluster is labeled to give information about the subtopic. If users select the cluster that contain results they are looking for, number of search results to be examined diminish logarithmically. Briefly, clusters and their labels guide users during their search experiences.

Now we consider the advantages of search result clustering. It provides an overlook of the search results. It also enables interaction with the user. Users could benefit from the clusters of search results by getting an overview of the query, selecting the cluster that is related to the question in mind or possibly changing the query according to the direction of a label. The results related to one subtopic are shown to the user in a compact view. In other words, interrelations between documents are revealed and results are presented by subtopics to the users. To summarize, advantages of SRC can be regarded as follows. Search result clustering:

- guides the user,

Figure 1.3: Main page of Bilkent news portal [7].

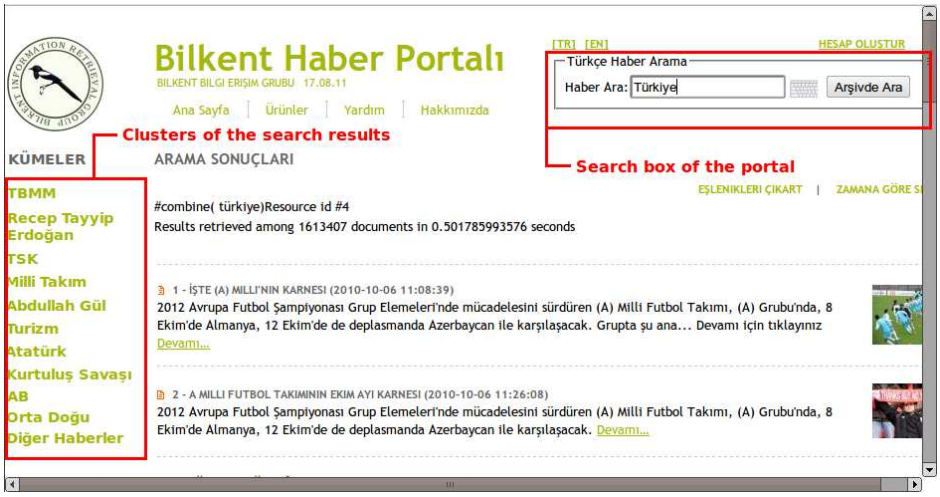

Figure 1.4: Bilkent news portal [7] shows the news related to query "Turkey" as a list. It also presents cluster labels on the left. (The image is tentative)

- gives an overlook of the results,
- provides an overview of the subtopics,
- improves interaction with user,
- helps user to reformulate the query, when needed,
- organizes search results by subtopics,
- decreases size of search results and search time in a logarithmic manner, when relevant results are found within a cluster.

According to the advantages described above, SRC especially suits to some type of search needs. It is beneficial, when users:

- enter a general query with diverse results, e.g. "computer science",
- enter an ambiguous query with a lot of meanings, e.g. "panther", exist in ambiguous entries provided by Wikipedia [3], possesses meanings from variety of subtopics like large cats, automotive, media, etc.,
- do not have sufficient information about the query,
- want to make a deep search about a topic. For instance, when a student is studying on a subject.

In fact, finding the underlying subtopics of search results returned for a query is a hard task. Even for people, manually clustering and labeling is a complex and time consuming work, so automatic solution of this problem is still open for improvement. Even though there exist a lot of search result clustering algorithms, embedding these methods into search engines is not a common practice. There are three main reasons behind this problem:

- Existing algorithms are not able to capture the relationships among documents successfully since the snippets are too short to convey enough information about query subtopics,

- Finding descriptive and meaningful labels for clusters is a difficult problem [28],
- The evaluation methodology is not well-defined for SRC task.

Motivated by these observations, we decide to work on this problem. We aim to enlighten this task a little more both from implementation and evaluation aspects Additionally, another motivation for this task is to embed the resultant search result clustering system into information retrieval service of Bilkent news portal. Main page of this portal is shown in Figure 1.3, that is constructed for research purposes. It employs the implementations of contemporary research topics like new event detection and tracking, duplicate detection, novelty detection, information retrieval and news categorization. After query "Turkey" is entered to the system, the portal outputs relevant news for the query, as shown in Figure 1.4. More information about Bilkent news portal can be obtained from [30].

1.2 Methodology

We present a new search result clustering method based on cover coefficient (C^3M) [8] and sequential k-means clustering algorithms [21]. We aim to cluster search results accurately by employing the powers of two linear time clustering algorithms. Most of the time, a combination of words, which is referred to *phrase*, is necessary to reflect the cluster content by conveying detailed information. We extract phrases using suffix tree data structure. Finally, we label clusters using a new method called "labeling via term weighting." This labeling scheme is based on term weighting as the name suggests and it prioritizes phrases found during the assignment of labels.

In addition, to evaluate the performance of the proposed method, we employ a comparative strategy. We estimate that such an approach for performance measurement can speed up the improvement of SRC task. We use two significant methods of SRC, namely suffix tree clustering (STC) and Lingo for comparative

performance evaluation of our method. In STC paper, it is proposed that clustering and labeling can be efficiently implemented through suffix trees. Lingo emphasizes the importance of labeling by firstly determining labels and then applying clustering using singular value decomposition. We have used implementations of these algorithms from Carrot^2 API [42].

We utilize some clustering evaluation metrics used in literature. The implementations of these evaluation metrics, namely, weighted f-measure, normalized mutual information, and contamination are also adapted from Carrot^2 API. On the other hand, for labeling evaluation, a new metric called similarity F-measure is presented which employ four different similarity metrics, namely, exact, partial, overlap and semantic similarity. For this part, we aim to give a new approach to automatic labeling evaluation, in addition to apply user based evaluations. Because the result of user assessments vary from person to person and hard to repeat for different parameters. It also prevents comparison of different methods.

1.3 Contributions

In this thesis, we

- design a new approach to search result clustering method, C^{3}M+K-means, based on C^{3}M and sequential k-means clustering algorithms. We adapt these two methods to the search result clustering problem with additional supportive steps: preprocessing, phrase extraction, and labeling.
- present a new labeling approach "labeling via term weighting" for assigning labels to clusters.
- introduce a new metric, $\text{sim}_{\text{F-measure}}$, by employing precision and recall, to assess the effectiveness of cluster labeling.
- propose to employ semantic similarity which is a research area of artificial intelligence for assessing the success of a label with respect to the ground truth label.

- present intuitive ways for determining similarity between ground truth label and label assigned by algorithm (in addition to semantic similarity metric): exact, partial, and overlap match strategies (i.e. similarity metrics).
- provide experimental results by systematically evaluating the performance of our method in the Ambient [12] and ODP-239 [13] test collections. We show that our method can successfully achieve both clustering and labeling tasks.
- adopt a comparative strategy for performance evaluation, using two prominent search result clustering algorithms: suffix tree clustering and Lingo.

1.4 Organization of Thesis

This thesis is arranged as follows:

- Chapter 1 introduces the search result clustering task. Also gives the motivation, methodology and contributions of this thesis.
- Chapter 2 presents the related background employed in the proposed method, specifically, generalized suffix tree and k-means clustering algorithm. In addition, related works about search result clustering task presented with special emphasis on suffix tree clustering and Lingo algorithms.
- Chapter 3 explains the proposed method, C^3M+K-means in detail.
- Chapter 4 focuses on performance measures used for clustering and labeling tasks.
- Chapter 5 introduces the Ambient and ODP-239 datasets. Additionally, it presents the performance results of the proposed method.
- Chapter 6 concludes this thesis with possible future pointers.

Chapter 2

Background and Related Work

In this chapter, firstly background about the methods that are used in our approach are presented. Afterwards, related works about SRC task are discussed with special emphasis on suffix tree clustering and Lingo algorithms that are used during comparative evaluation in Chapter 5.

2.1 Background

As background information, suffix tree is introduced, which is used for phrase extraction in Section 3.2. K-means is discussed because its modified version, sequential k-means is used for clustering in Section 3.5.2.

2.1.1 Generalized Suffix Tree

Suffix tree introduced by [18, 40] is a rooted, directed, compact tree data structure, holding all suffixes of a string. This data structure is used in a variety of research areas. For example, it enables very fast and memory-efficient comparison of the genomes [15], so that it is employed in bioinformatics applications.

Generalized suffix tree, is a type of suffix tree structure with multiple strings instead of one inserted into the tree. In this thesis, we will use the terms 'suffix tree' and generalized suffix tree interchangeably. Instead of character-level insertion to the suffix tree, word-level suffixes of texts are added. It is similar to inverted index [47] and can be used as an indexing structure. In addition, It is used during clustering and phrase exploration processes, firstly by Zamir et al. [43]. It can be constructed in linear time with number of documents using Ukkonen's algorithm [39]. In our method, we use suffix tree for finding phrases existing in the snippets, similar to [43].

There are four types of nodes in a suffix tree:

1. **Root node** is the only node with no parents and called as the *root* of the tree.
2. **Edge nodes** are invisible nodes on the edges and they hold a label.
3. **Internal nodes** are the nodes with more than one children.
4. **Suffix nodes** are the nodes that contain at most one child node. They designate the information about the suffix. In detail, the information about suffix, sentence and document.

An edge is labeled using an edge node and each suffix can be regenerated by labels of edge nodes from the root to a leaf. Each node represents a label and documents that contain its label. The information about documents are gathered from the children suffix nodes of the node. And the information about its label is obtained from the combination of labels of edge nodes starting from root to itself.

Phrases are more informative than single-words, so they are better candidates for labeling. Phrase discovery is a crucial phase for SRC task, which aim to generate meaningful cluster labels. Suffix tree structure indexes the sequence of words in the nodes and stores number of occurrences of them. The inner nodes with sufficient occurrences are considered as frequent phrases. For phrase

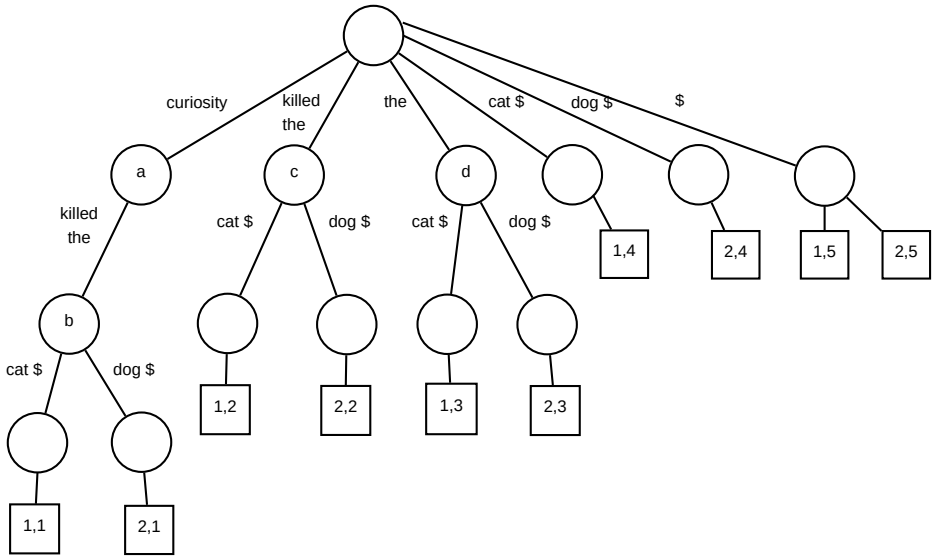

Figure 2.1: Generalized suffix tree example for two documents (without stopword elimination and stemming): 1) Curiosity killed the cat. 2) Curiosity killed the dog. The $ sign is added to the end of sentences, to mark the end of the sentence.

extraction, we use the labels of internal nodes with more than two documents. More information about phrase extraction (discovery) can be found in Section 3.2.

Figure 2.1 demonstrates a suffix tree obtained from two documents. The nodes that appear in at least two documents are labeled from a to d. The suffix nodes contain at least one rectangle box that store the suffix information. For instance the leftmost suffix is "curiosity killed the cat", which exists only in the first document's first suffix (itself), so it is attached to the box named as "1.1".

2.1.2 K-means Clustering Algorithm

K-means is a linear-time and widely used clustering algorithm which groups given inputs into partitions. It is introduced in the works of Forgy and Rocchio [17, 35] circa 1965. It operates in an iterative manner to refine clustering structure, until it reaches the criteria defined for stability. This is generally the convergence to a total minimum squared error (also named as total within-cluster variation).

General working principle of k-means is provided in Algorithm 1. As seen from the algorithm, k-means starts after the initial centroids are provided, or it can also start without centroids using randomly selected documents as centroids. Each data is assigned to the cluster with closest centroid using euclidean distance. The disadvantage of k-means is that it requires spherical cluster structures.

Algorithm 1 K-means Algorithm

if Input: Number of clusters (n_c) and initial centroids of clusters **then**
  Assign input centroids as the centroids of empty clusters
end if
if Input: Number of clusters **then**
  Select n_c documents randomly from document collection
  Set selected documents as centroids of empty clusters
end if
for Total minimum squared error is greater than threshold **do**
  for Each document in the collection **do**
    Assign document to the cluster whose centroid is closest to the document
  end for
  for Each cluster **do**
    Update the cluster centroid using documents of the cluster
  end for
end for
Output: clustered documents

2.2 Related Work

In this section, the related works on SRC are presented via the two prominent algorithms: Lingo [31] and Suffix Tree Clustering [43]. These are the two methods which we use during comparative evaluation of our algorithm. Additionally, some of the significant works on search result clustering are covered. In this thesis we use post-retrieval clustering, web document clustering, and search result clustering interchangeably.

2.2.1 Suffix Tree Clustering (STC)

Suffix tree clustering is proposed by Zamir and Etzioni in [43] for web document clustering. It employs phrases obtained from suffix tree for clustering and labeling tasks. According to Zamir et al. search result clustering problem is suggested to have the features presented below:

1. **Relevance**: Each cluster should contain relevant documents with each other.
2. **Browsable Summaries**: Cluster labels should be concise and descriptive, so that they are easy to understand at a glance.
3. **Overlap**: Documents which contain multiple topics could appear in more than one cluster
4. **Snippet-tolerance**: Due to the time limit, search result clustering should depend on snippets.
5. **Speed**: The method should be fast enough not to delay the search operation of users. It is preferred to be a linear time algorithm.
6. **Incrementality**: The method should incrementally process results as they are provided by the search engine.

In this list, the first item *relevance* corresponds to clustering quality and second *browsable summaries* represents labeling quality throughout the thesis.

STC is a linear time clustering algorithm based on identifying common phrases to all documents. A phrase is defined as an ordered sequence of one or more words. Its difference from other clustering algorithms is that STC considers a sentence as a sequence of connected words instead of common bag of words usage. It makes clustering and labeling using common phrases between documents using suffix tree data structure, that we present in Section 2.1.1. STC consists of three main steps: document cleaning, base cluster identification using a suffix tree, and constructing final clusters through combining base clusters.

First of all, noisy data is filtered out from text documents for a successful clustering process. Therefore, the first step is preprocessing which includes sentence boundary identification, text cleaning (to remove non-word tokens like numbers, HTML tags, etc.), and lastly stemming.

Second step is the identification of base clusters. During this process, firstly suffix tree structure is constructed. Precisely, suffixes of each sentence in the document snippets are fed to the suffix tree. This operation is similar to creating an inverted index of terms for the document collection. We discuss suffix tree broadly in Section 2.1.1. In this step, all nodes of suffix tree are treated as cluster candidates. Then, internal nodes whose labels appear in at least two documents are chosen to be the base clusters of the document collection. At the end of this step, each base cluster is assigned a score which is a function of the number of documents it appears and number of words that make up the phrase.

Finally, the third step is final cluster formation where the base clusters are combined according to their documents to avoid nearly identical clusters. This step begins with determining the similarities between base clusters. Two base clusters are assumed to be similar if they share at least half of their documents with each other. Then, similar base clusters are combined into a single larger cluster to form a final cluster. The number of final clusters is generally high. Therefore, final clusters are assigned scores based on their base clusters and their overlap. Then, the topmost 10 clusters are selected to be presented to the user. Each cluster's label is set to the concatenation of its base clusters' labels. Note that, as the post-retrieval clustering evolves, it is accepted that each label contains only one term (single-word or phrase) instead of several terms presented consecutively. The reason for this approach is to enable readability of labels at a glance.

2.2.2 Lingo: Search Result Clustering Algorithm

Lingo is a search result clustering algorithm based on singular value decomposition (SVD) [31]. The algorithm emphasizes cluster description quality (labeling

quality) which is an important factor for human-friendly search engines. To increase the labeling performance, Lingo reverses the current search result clustering procedure where clustering is followed by labeling. Lingo first reveals cluster labels and then assigns documents to these labels. The Lingo method is currently being used in Carrot^2 Open Source Search Results Clustering Engine [42].

Lingo algorithm consists of the following steps *preprocessing*, *frequent phrase extraction*, *cluster label induction* and *cluster content discovery*. The following parts explain these steps briefly [31].

Preprocessing: The aim of the preprocessing phase is to clean the input documents from all characters and terms that can possibly effect the quality of cluster descriptions. Although Lingo uses SVD, which is capable of dealing with noisy data, the preprocessing phase is still required to avoid meaningless frequent terms in the cluster labels. There are three main steps in preprocessing phase; text filtering to remove HTML tags, entities and non-letter characters, language identification and finally stemming and stopword removal.

Frequent phrase extraction: The frequent phrases are defined as the recurring ordered sequences of terms in document snippets. These phrases are chosen as candidate cluster labels if they:

1. appear in document snippets more than *term frequency threshold*
2. stay in sentence boundaries
3. are a *complete phrase* (definition can be found in [31])
4. do not begin nor end with a stopword.

Cluster label induction: The cluster label induction step is based on the extraction of frequent terms (including single word terms) and consists of three main steps; term-document matrix construction, abstract concept discovery and label pruning.

In the first phase, the term-document matrix is constructed with the single word terms that exceed the term frequency threshold. Then, weight of each term

is calculated by using the *term frequency, inverse document frequency (tfidf)* formula [36].

Once the term-document matrix is constructed, the abstract concept discovery phase begins. In this step, singular value decomposition is used to find the orthogonal basis of the term-document matrix. In the paper, it is claimed that these orthogonal basis, at least hypothetically, corresponds to abstract concepts appearing in the original term-document matrix. SVD breaks the vector space matrix A with t terms and d documents into three matrices,

$$A = U\ \Sigma\ V^T \tag{2.1}$$

where U and V are left and right singular vectors of matrix A and Σ contains singular values diagonally. The important thing here is to notice that, only the first k vectors of the U matrix are used, meaning that only the first k abstract concepts will be investigated for cluster label candidacy.

Finally, label pruning steps begins to determine the cluster labels. The important thing to notice in this step is that both the abstract concepts and the frequent phrases are expressed in the same vector space-the column space of the original term-document matrix A. Thus, the classic cosine distance is sufficient to determine the closeness of phrases to abstract concepts. Currently, we have t frequent single word terms and p frequent phrases as the candidate cluster labels and k abstract concepts which require a human readable cluster label. To match the abstract concepts with the frequent phrases, we first build a $t \times (t + p)$ P matrix by treating the phrases and keywords as pseudo-documents. Then, the closeness is to i^{th} abstract concept is calculated as $m_i = U_i^T P$ where U_i is the i^th column of the U matrix. After that, the phrase that corresponds to the maximum component of the vector m_i is selected as the human-readable description of the i^th abstract concept. To extend this methodology to complete U_k matrix, they generate an M matrix as $M = U_k^T P$ which yields the results of all abstract concept, frequent phrase pairs. Finally, a label pruning is used to prune overlapping label descriptions whose details can be found in [31].

Cluster content discovery: In this phase, all the documents are re-queried to be assigned into previously determined cluster labels. This assignment is

achieved with the classic Vector Space Model. First each cluster label is represented as a column vector forming a matrix of labels, Q. Then, $C = Q^T A$ indicates the strength of membership of the documents to the cluster labels. A document is assigned to a cluster if c_{ij} exceeds the *snippet assignment threshold*. The documents not assigned to any cluster end up in an artificial cluster called *Others*.

2.2.3 Other Works on Search Result Clustering

The utility of search result clustering and associated cluster labeling algorithms for easy access to the query results has been widely investigated [11]. The background of the research dates back until 1990s [14] [4] [20] [25]. More recently; however, there are continuous research and commercial efforts for developing online search result clustering and labeling methods [11].

Prior to this method, there has been an extensive research on search result clustering but the very first results were introduced in the Scatter-Gather system [20]. Consecutively, Suffix Tree Clustering and Lingo [31] are prominent works about SRC task. Apart from those, MSEEC [19] and SHOC [45] also contribute the use of words proximity in the input documents. A comprehensive review of research done about search result clustering is presented in a survey paper by Carpineto et al. [11].

Kural et al. [23] employ cover coefficient clustering for search result clustering in 2001. They do not use k-means clustering for the refinement of clusters after C^3M and they show each cluster with three representative documents and ten terms (with nearly all of them are single-word terms). User studies show that the users are not satisfied with the cluster based presentation of search results. In parallel with this study, nowadays, the inclination is towards using only one term to represent a cluster, as in our study, not to distract users with a lot of information.

Moreover, clustering web results is also essential for mobile devices since it

decreases the amount of information transmitted, provides a more effective and informative user interface. Therefore this type of interface requires less interactions in terms of page scroll or query reformulation [10] [11].

Furthermore, there exists another approach to enhance users' search experience called search result diversification. This research topic aims to re-rank search results for presenting documents from different subtopics at the beginning of search results [9]. This research area is also based on post-processing of search results like SRC.

Chapter 3

Search Result Clustering and Labeling

The methodology we use in this study is to extract the relationships among documents with C^3M method and to construct the final clusters through feeding the results of C^3M to the sequential k-means algorithm. Then we use term weighting-based approach to label the generated clusters. In this chapter, we describe the details of the proposed method, which is composed of the following steps:

- Preprocessing and vocabulary construction
- Phrase discovery
- Term weighting
- Indexing
- Cover coefficient clustering
- Sequential k-means clustering
- Labeling via term weighting

Figure 3.1 illustrates the processes we adopt in this study for our SRC method. As shown in the figure, user enters query "EOS" to the clustering based search

engine and results are presented to the user as labeled clusters: “Wolkswagen EOS”, “EOS Electronic” and “Digital Camera”. This example query is taken from the Ambient dataset, and the labels are suggested by the proposed method for this query.

3.1 Preprocessing and Vocabulary Construction

Our method starts with preprocessing which is a common phase in all IR problems. Preprocessing is applied to eliminate the redundant data in document snippets and convey informative features for the clustering and labeling processes. For a query, the search results are provided to the proposed method, where each result consists of a URL, title and a very short text. We form the document text by concatenating title and short text. Then, these document texts are cleaned from non-letter characters and uppercase letters are converted to lowercase. Words that contain punctuation mark within their boundaries, are converted into separated words. To illustrate, “cluster-based” is converted into two words: “cluster” and “based”.

At this point, tokenization starts, which aims to separate text into words. For this purpose spaces within text are used to determine words. Afterwards, words that occur in stopwords list are eliminated. Stopword removal is important because these words can appear in any text and they are not informative for clustering and labeling (e.g. and, after, we). And stemming is applied by the Porter Stemmer [33] to treat words which have common stem as the same feature. Finally, all document texts are combined for selection of terms which are estimated to convey information for differentiating documents from each other. In addition, selection process increase the performance by decreasing the size of vocabulary. Precisely, the terms whose number of occurrences are between 3-30% of the number of snippets are selected to construct the vocabulary of document collection. Note that, after we determine the phrases in the subsequent section, these phrases are also inserted into the vocabulary.

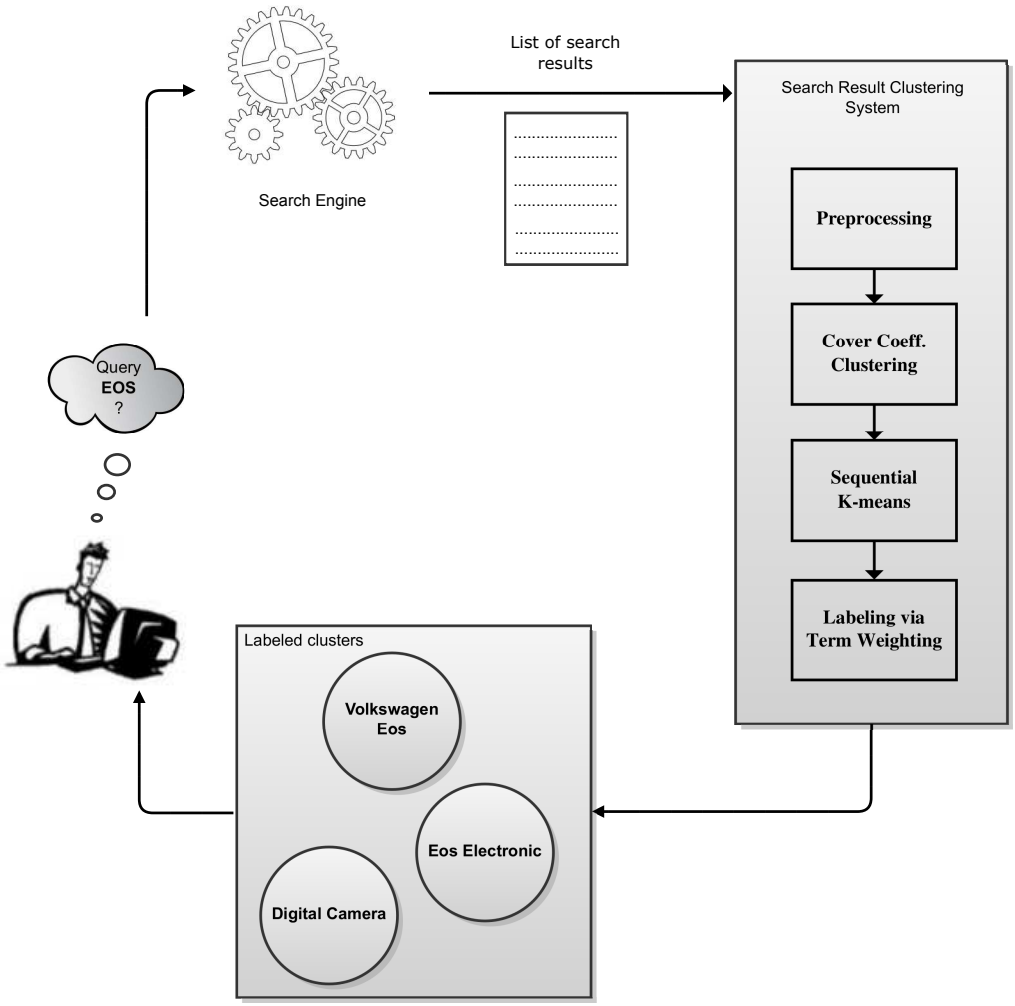

Figure 3.1: Search result clustering processes of the proposed method

Note that for the construction of suffix tree in Section 3.2, we perform a different preprocessing phase where we also keep the punctuation marks (dot, exclamation mark, question mark, semicolon, colon) which define the sentence boundaries. Semicolon and colon are also used as sentence boundaries because they are generally used for the combination of two sentences. By this way, sentences are separated from text and inserted into the suffix tree. In general, stopwords are kept for suffix tree construction [43]. But, we remove stopwords to obtain more generic phrases.

In order to regenerate original labels from preprocessed versions, mappings of each preprocessed term's position in preprocessed text to its corresponding position in original text are stored.

3.2 Phrase Discovery

For descriptive cluster labels, the phrases are more informative than words, as we have discussed in Section 2.2. Therefore, we find the common phrases that are good candidates for labels. In addition, phrases possess more discriminative value for clustering of documents than single-word terms. Therefore, we treat the extracted phrases similar to single terms found in Section 3.1 by adding them to the vocabulary.

However, the discovery of such phrases from plain text is a difficult task since they should contain meaningful word combinations to be a good cluster label. In this study, we use suffix tree structure [43] to extract frequent phrases from document snippets. Suffix tree indexes sequence of words in the nodes and stores number of occurrences. We describe suffix tree as background knowledge in Section 2.1.1. The inner nodes with sufficient appearances in different documents are considered as phrases and added to the vocabulary. Precisely, labels of nodes that occur in more than %2 of the documents are selected.

3.3 Term Weighting

Term weighting is employed for determining the importance of a term for a document. Weight increases as term occurs more in document, but decreases as it appears more in collection. The term weights are computed by using the log entropy formula in [16]. This weighting scheme is commonly used for latent semantic indexing (LSI) based methods [24]. It can be described as:

$$F_{ij} = L_{ij} G_j \tag{3.1}$$

which defines the weight of j^{th} term in i^{th} document where L_{ij} and G_j represents respectively the local and global weights of j^{th} term. Local weight represents the importance of term within the document, and global weight reflects the affect of term's occurrences in the collection. These weights are computed by using the following formulas.

$$L_{ij} = \log_2(tf_{ij} + 1) \tag{3.2}$$

$$G_j = 1 + \frac{H_j}{\log_2(m)} \tag{3.3}$$

$$H_j = -\sum_{i=1}^{m} p_{ij} \log_2(p_{ij}) \tag{3.4}$$

$$p_{ij} = \frac{tf_{ij}}{\sum_{i=1}^{m} tf_{ij}} \tag{3.5}$$

where m is the number of documents and tf_{ij} is the term frequency (how many times term occurs in document).

As seen in the formula, local weight is calculated using the log function, where adding one provides nonnegative output. Using logarithm of term frequency reduces the affect of high differences between term occurrences. Global weight is calculated by normalized entropy, where normalization is obtained by the division in global weight. This equation includes one added to normalized entropy, to obtain nonnegative output. p_{ij} represents the term frequency normalized by the total number of occurrences of term in collection.

Entropy is a concept of information theory, shows the deviation from uniform distribution. H_j denotes the entropy of j^{th} term. Entropy based global weighting is one of the most sophisticated global weight calculation methods which considers the distribution of term over documents. This scheme gives lower weight to frequent terms and higher weight to infrequent terms. Entropy measures the uncertainty, so smallest entropy is obtained when all values are equal. If entropy is high, this means the weights of term in different documents shows high fluctuation, and such a term is informative with high global weight.

Finally, we increase the weight of phrases with respect to single-word terms, by multiplying phrases with a constant value θ and single-word terms with $1-\theta$ as shown in equation 3.6. In our experiments, the best results are acquired when θ is 0.7. Due to the re-weighting operation, normalization of term weights in each document is required.

$$F_{ij} = \begin{cases} F_{ij} \times (1-\theta) & \text{if } |j| = 1 \\ F_{ij} \times \theta & \text{otherwise} \end{cases} \tag{3.6}$$

3.4 Indexing

Before passing to the clustering phase, we index each document using the terms it contains. Weights of terms are calculated using term weighting scheme provided in Section 3.3. Only the terms in vocabulary are kept in the indexing data structure. We employ forward indexing approach for document representation where each document holds the terms it contains [26]. In Figure 3.1, a simple representation of forward indexing is demonstrated.

Using forward indexing is advantageous for SRC task and datasets we use, because we have about 100 documents for each query and few number of terms for each document. We prefer forward indexing because, clustering and other operations of the proposed method requires the comparison of two documents, or document based information. These operations are efficiently handled using

Table 3.1: Forward indexing example for three documents (without stopword elimination and stemming): 1) Curiosity killed the cat. 2) Dog killed the cat too. 3) Curiosity killed the dog too.

Document	Index
Document 1	cat, curiosity, killed, the
Document 2	cat, dog, killed, the, too
Document 3	curiosity, dog, killed, the, too

forward indexing.

Other indexing methods are inverted index and vector space indexing as shown in Figure 3.1 and Figure 3.3 for comparison. Vector space model stores the documents as a sparse matrix with most of the indexes are empty. Inverted index is more space-efficient than vector space indexing. It stores mappings of terms to the documents that each term appears. While inverted index employs term-based storage, forward indexing uses document-based storage. In addition, inverted index is more compact than forward index, so it is appropriate for large document collections. Search engines firstly create the forward index of a new document and transform it to be inserted into the inverted index.

We do not employ inverted index, which is a widely used indexing structure in IR. It applies indexing according to terms of collection instead of documents in forward index. Inverted index is useful for information retrieval task that finds relevant documents for query terms. In fact, forward indexing combines the features of inverted index and vector space indexing as being space-efficient and document-based storage. We use forward index instead of inverted index because it is more suitable for comparison of documents in small datasets.

Table 3.2: Inverted indexing example for three documents (without stopword elimination and stemming): 1) Curiosity killed the cat. 2) Dog killed the cat too. 3) Curiosity killed the dog too.

Document	Index
cat	Document 1, Document 2
curiosity	Document 1, Document 3
dog	Document 2, Document 3
killed	Document 1, Document 2, Document 3
the	Document 1, Document 2, Document 3
too	Document 2, Document 3

Table 3.3: Vector space indexing example for three documents (without stopword elimination and stemming): 1) Curiosity killed the cat. 2) Dog killed the cat too. 3) Curiosity killed the dog too.

Term	Document 1	Document 2	Document 3
cat	√	√	
curiosity	√		√
dog		√	√
killed	√	√	√
the	√	√	√
too		√	√

3.5 Clustering

3.5.1 Cover Coefficient Clustering (C^3M)

Cover coefficient clustering is a seed oriented, partitioning, single-pass, linear-time clustering algorithm introduced in [8]. The main goal of C^3M is to convey the relationships among documents using a two-stage probability experiment. This experiment is used for computation of similarity between d_i and d_j, which is the probability of obtaining d_j from d_i as illustrated in figure 3.2. The efficiency and effectiveness of C^3M in texts is experimentally demonstrated in [6] for information retrieval. Cover coefficient concept is mainly used for:

- Identifying relationships among documents,
- Deciding number of clusters to be generated,
- Selecting seed documents,
- Forming clusters through grouping non-seed documents around seed documents.

We first initialize the C matrix which conveys the document to document relations with size mxm, where m is the number of documents in the collection. Index at i^th row and j^th column of C matrix; c_{ij} represents the extent to which document i is covered by document j, in other words coupling or similarity of d_i with d_j (where d_i represents the $i^{th} document$). Each element of the C matrix is calculated as

$$c_{ij} = \alpha_i \times \sum_{k=1}^{n} (F_{ik} \times \beta_k \times F_{jk}) \qquad 1 \leq i, j \leq m \tag{3.7}$$

where α_i and β_k represent the reciprocals of the term weight sum in i^{th} document and in collection, respectively. They are used for normalization and formulated below, where n represents number of terms:

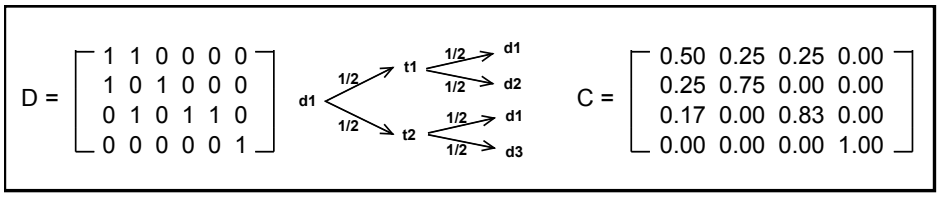

Figure 3.2: C^{3}M described: D matrix (document-term matrix with m=4, n=6), two stage probability experiment for the first document in the middle and C Matrix shown

$$\alpha_i = [\sum_{j=1}^{n} F_{ij}]^{-1} \quad 1 \leq i \leq m \tag{3.8}$$

$$\beta_k = [\sum_{j=1}^{m} F_{jk}]^{-1} \quad 1 \leq k \leq n \tag{3.9}$$

In Figure 3.2, C^{3}M is described with the computation of C Matrix. Note that, the C matrix does not need to be constructed completely but only the required indexes are computed. Besides, the original C^{3}M algorithm also computes the number of clusters to be generated [8], however, we assign the documents into 10 clusters with an additional *Others* cluster to store the outlier documents.

The next step in our modified C^{3}M algorithm is to find the seed documents of the clusters. Seed documents must be well separated from each other and must have the capability of attracting the non-seed documents around themselves. Therefore, the concept of *seed power* is introduced in [8] to satisfy such conditions where seed power of the i^{th} document can be calculated as follows:

$$P_i = \delta_i \times \psi_i \sum_{j=1}^{n} (F_{ij} \times \delta'_j \times \psi'_j) \tag{3.10}$$

$$\delta_i = c_{ii} \tag{3.11}$$

$$\psi_i = 1 - \delta_i \tag{3.12}$$

In these equations δ_i represents the decoupling which we define as the uniqueness of the document from the rest and ψ_i represents the coupling which is the measure

of how much the document is related to other documents in collection. Finally, the summation part in Equation 3.10 ensures the normalization. Now the only unknowns in Equation 3.10 are δ'_j and ψ'_i which define the concepts for terms and they are the counterparts of the metrics δ_j and ψ_i we define above for documents. c'_{ij} denotes the relationship of i^{th} term with j^th term. They can be computed by combining the below equations.

$$c'_{ij} = \beta_i \times \sum_{k=1}^{m} F_{ki} \times \alpha_k \times F_{kj} \tag{3.13}$$

$$\delta'_i = c'_{ii}, \tag{3.14}$$

$$\psi'_i = 1 - \delta'_i, \tag{3.15}$$

Then, we choose 10 documents with the topmost seed powers as the seed documents for our clusters. We apply a special case if seed powers of two documents in the top 10 documents are very close to each other to eliminate false (similar) seeds. In such cases, we ignore one of the documents and choose another seed document from the collection according to the seed power.

The final part of C^3M document clustering is to form final clusters by assigning the nonseed documents to the clusters which are defined by a seed. To accomplish this task, for each nonseed document we check the coverage of the document with the seed documents (from C matrix) and select the seed that has the highest coverage over the nonseed. If none of the seeds cover the nonseed document, then, this document is directly added to the *Others* cluster. More detailed information about C^3M can be found in [8].

3.5.2 Modified Sequential K-means Algorithm

We introduce k-means algorithm as background information in Section 2.1.2. It is a linear-time and widely used clustering algorithm which groups given documents after the initial centroids are provided. The success rate of the k-means algorithm depends on the initial cluster centroids. Therefore, we use the results of C^3M clustering to derive the centroids as accurately as possible. The input centroids

are the vectorial averages of the documents in each C^{3}M cluster. For convergence, we expect the total minimum squared error (total within-cluster variation) is below a threshold.

We use a variation of k-means, called sequential k-means algorithm [21, 27] that updates the cluster centroid after each document assignment to the cluster instead of after all documents are distributed in original k-means [17, 35]. We use a modified version of the sequential k-means algorithm where we assign documents to the centroids as in original k-means in the first iteration. And, centroids are re-calculated according to the new distribution of documents. At the beginning of the following passes, we empty the cluster contents. Then, we assign each document to the nearest cluster and update that cluster's centroid again after this assignment using the formulation below.

$$centroid_i = \frac{\sum_{j \epsilon cluster_i} doc_j + centroid_i}{|cluster_i| + 1} \tag{3.16}$$

where $|cluster_i|$ is the number of documents in the cluster and 1 is added to the denominator for the centroid vector in numerator. Other properties of sequential k-means are same as original k-means defined in Section 2.1.2. Algorithm 2 describes the working principle of sequential k-means.

The first iteration is the same as original k-means iterations. This approach increases effectiveness by firstly stabilizing the centroids to some extent. For instance, sometimes the centroids may be given randomly to sequential k-means, so computation of centroids at the beginning is advantageous.

The drawback of sequential k-means is that it is order-dependent. The documents are assigned to clusters one by one and after each assignment the cluster's centroid is updated using the newly added document. The order-dependency comes from the order of documents separated into clusters. This disadvantage is weakened by using the centroid itself in Formula 3.16 during the update of centroid.

Algorithm 2 Sequential K-means Algorithm

Input: number of clusters and initial centroids of clusters
First iteration:
for Each document in the collection **do**
  Assign document to the cluster whose centroid is closest to the document
end for
for Each cluster **do**
  Update the cluster centroid using documents in the clusters
end for
Next iterations:
for Total minimum squared error is greater than threshold **do**
  Empty the clusters
  for Each document in the collection **do**
    Assign document to the cluster whose centroid is closest to the document
    Update the centroid to which document is assigned using Formula 3.16
  end for
end for
Output: clustered documents

3.6 Labeling via Term Weighting

The final step of our method is the labeling phase. We aim to assign descriptive labels to clusters in order to reflect the content of the cluster.

There are two cluster labeling strategies [28]:

- **Cluster-internal labeling:** It labels the cluster by considering terms in the cluster.
- **Differential cluster labeling:** In this labeling strategy, both the terms in the cluster and their behavior in the collection are considered.

If a term occurs frequently in the cluster and also a common term in the collection, then it is not suggested by the second strategy. In literature, feature selection methods such as information gain, mutual information and X_2 are offered for employing differential cluster labeling.

We present a novel labeling strategy called *labeling via term weighting* which

adopts the approach of differential cluster labeling. It is based on term weighting approach used in information retrieval. Term weighting is used to determine the importance of term form the document. Therefore, we can also use term weighting for calculating the significance of term for cluster.

Firstly, the terms of all documents in a cluster are merged. For this aim, we combine the number of occurrences of terms that appear in documents of the cluster, to make the effect of combining original texts of documents in the cluster and assuming the cluster as a document with the combined text. Then, term weighting is applied to the clusters, by assuming them as documents. The same term weighting formula is used as in Section 3.3. A single-word label generally lacks expressiveness, so we give more weight to phrases than single-word terms during cluster labeling as in Section 3.3. Now, each term occurring in cluster has term weight assigned to it, denoting the importance of term for the cluster.

For each cluster, we select the highest weighted terms into the candidate labels list. In our experiments, we add topmost five terms to the list. While we are assigning the final label of the cluster from this list, we follow the criteria below:

- Clusters are labeled in descending order of cluster size,
- Label should not be one of the previously given labels to another cluster,
- Phrase label candidate with less than five words is preferred (if exists),
- Terms that are in higher ranks of the list (have higher weights) are preferred.

In this step we select labels for these clusters. However, these labels are preprocessed, particularly, cleaned, stopwords removed and words are stemmed. Therefore, we obtain the original versions of the preprocessed labels using positions of preprocessed text and respective positions in original text that are recorded during preprocessing step in Section 3.1.

The research presented in this thesis is submitted for publication [38].

Chapter 4

Performance Measures

In this chapter, the evaluation metrics that are employed for assessing the success of clustering and labeling tasks are described. The datasets used during evaluation provide the ground truth (gold standard) clusters and their labels for each query. Ground truth cluster is mentioned as "class" throughout the thesis. We use the ground truth information to assess the success of the proposed method. The more proposed method's output resembles to the ground truth from clustering and labeling aspects, the better performance is reached. We use a comparative strategy to derive the relative performance of our algorithm with respect to the two state-of-the-art algorithms: Lingo and suffix tree clustering. Implementation of these methods are available in Carrot^2 API [42].

4.1 Clustering Evaluation

To be able to quantify the performance of the algorithms in a common way, we first need to define success measures which reflect the actual performance of clustering results as fairly as possible, regardless of the clustering method we choose.

4.1.1 Evaluation with Random Clustering

The first step of the clustering evaluation is to prove that the algorithm shows significant difference from random clustering. For this goal, we firstly use the Monte Carlo method [21]. If the cluster sizes are preserved and documents are distributed to the clusters randomly, we obtain a random clustering. The definition of target cluster for a class, is the cluster that contains at least one common document with the class. Figure 4.1 demonstrates the target clusters for each class for an example clustering structure. As a rule, in a meaningful clustering structure, the average number of target clusters of the clustering method (represented as n_t) should be less than the average number of target clusters of random clusterings (represented as n_{tr}) [8]. For each query, 1000 random clusterings are generated to stabilize the random behavior of n_{tr} value.

We also use t-test for proving the statistical significance of the difference between random and proposed method. T-test is a way of statistical hypothesis testing, which determines whether a result is statistically significant, meaning that it does not occur with chance according to a threshold probability. We compare two distributions of samples obtained from queries of dataset for random clusterings and algorithm generated outputs. One sample for each distribution is n_t and n_{tr} values for a query. The data is provided in Table A.1 (for the Ambient dataset) and Table A.2 (for the ODP-239 dataset) in Appendix. Our aim for applying t-test is to prove that the proposed algorithm performs significantly different than random. Therefore, we use paired difference test to compare two sets of measurements to determine whether the sets' population means are different. Paired t-test is used for proving the statistical significance with respect to a threshold value, 0.01 in our tests. If the significance value calculated between distributions are below the threshold, then statistical significance is assured.

Additionally, we use t-test experiments to convey the significance of one method with another to prove that the success of the first method is unlikely to occur by chance. During these experiments, we use the same type of t-test throughout the thesis with significance threshold set to 0.01.

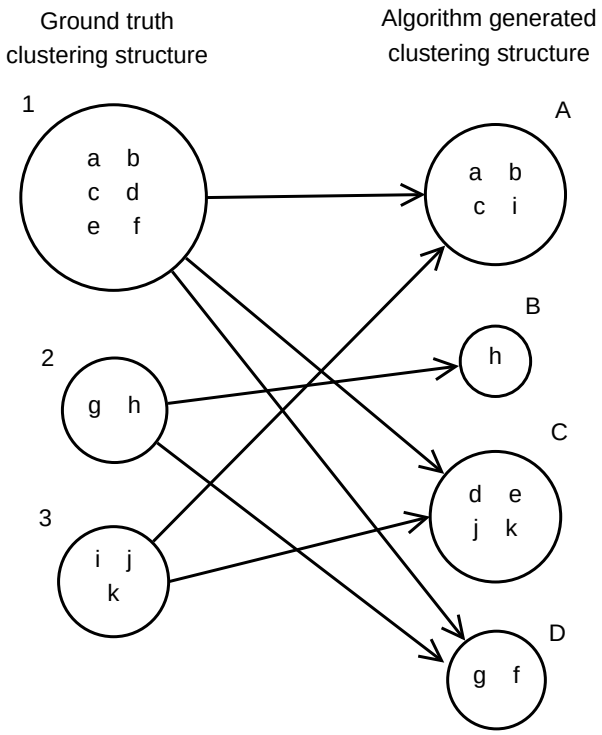

Figure 4.1: Demonstration of target clusters for the ground truth classes. For class_1 target clusters are cluster_A, cluster_C, cluster_D, because the documents in this class are separated into these clusters. Average number of target clusters for the algorithm generated clustering structure, n_t, is 2.33. This value is calculated by summing up the target clusters of all classes and diving by the number of classes (average number of target clusters is calculated as follows: $3 + 2 + 2 = 7$. Then, we take the average of this sum: $7/3 = 2.33$).

4.1.2 Main Evaluation Measure: $w_{F\text{-measure}}$

Now, we continue with the main evaluation measure we use for assessing the success of clustering with respect to the ground truth. In this thesis, we use the weighted average F-measure (in short weighted F-measure or $w_{F\text{-measure}}$) defined in [41] which is the average of total weighted F-measure of each class.

This evaluation measure is based on measuring how successfully a class is represented by a cluster. This is achieved by precision and recall metrics between a class and a cluster. Intuitively, precision reflects to what extend presented cluster includes documents of ground truth class and recall reflects to what extend ground truth class is presented to the user.

For calculating the success between a class and a cluster, we need to match the class with one of the clusters that give the highest F-measure with the class. Hence, we find the representative of class from the clusters generated. In Figure 4.2, matching classes and clusters are provided as an example.

The similarity between a ground truth class i and represented (matching) cluster j are calculated as follows.

$$precision(i,j) = \frac{class_i \cap cluster_j}{|cluster_j|} \tag{4.1}$$

$$recall(i,j) = \frac{class_i \cap cluster_j}{|class_i|} \tag{4.2}$$

$$F\text{-}measure(i,j) = \frac{2 \times recall(i,j) \times precision(i,j)}{recall(i,j) + precision(i,j)} \tag{4.3}$$

We find the precision and recall values as shown in Equations 4.1 and 4.2 to obtain F-measure in Equation 4.3 related to the class i and matching cluster j. F-measure is the harmonic mean of precision and recall. F-measure is used for the calculation of $w_{F\text{-measure}}$ described in Equation 4.4. It is also employed for labeling evaluation in Equations 4.5 and 4.6 to match classes and clusters. It shows the partial success of clustering, by measuring how successfully a class is represented in clustering.

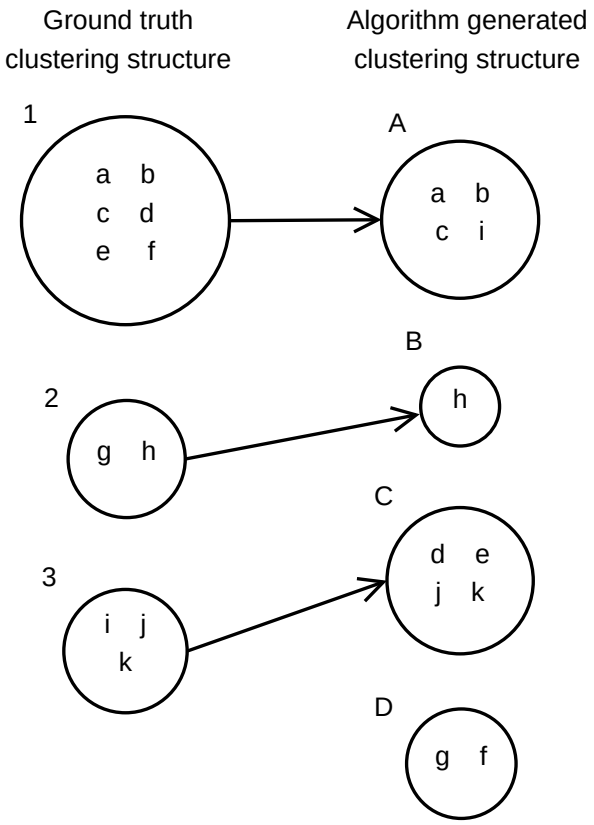

Figure 4.2: Demonstration of matching classes and clusters for the calculation of $w_{\text{F-measure}}$ using Equation 4.4 for example clustering structures. A representative cluster is found for each class, which gives the highest F-measure among all clusters. (For instance, class_1 shares common documents with cluster_A, cluster_C, and cluster_D. But, F-measure between this class and cluster_A is higher than other cases. With precision 3/4, recall 3/6 and their harmonic mean, F-measure is 0.6.)

Now, we have the success scores of class representations on the algorithm generated side. We combine these success scores of class representations by weighting with class size, summing up and taking the average. And we obtain the weighted average F-measure, $w_{\text{F-measure}}$, overall success of clustering. Being composed of the harmonic mean of precision and recall, $w_{\text{F-measure}}$ evens up the affect of number of clusters. Overall clustering performance is computed as follows.

$$w_{F\text{-}measure} = \frac{1}{\sum_{i=1}^{n_{class}} |class_i|} \sum_{i=1}^{n_{class}} (\max_j \{F\text{-}measure(i,j)\} \, |class_i|) \quad (4.4)$$

where n_{class} and $n_{cluster}$ are respectively the number of classes and clusters. If we summarize, we compute the success score, F-measure, between each ground truth and represented cluster with Equation 4.3. Then, for each class in the ground truth we find the best matching cluster (that has the maximum F-measure among all clusters) is selected. The maximum F-measure for each class is multiplied with class size to add weight. And lastly, the values obtained are summed up and the average is the resultant success of the generated clustering.

4.1.3 Evaluation with Supportive Measures

Additionally, we use purity, contamination [31] and normalized mutual information (NMI) evaluation metrics during the evaluation of algorithm generated clusters with ground truth.

- **Contamination:** Cluster contamination measure described in [31], aims to assess how much clusters are contaminated by containing documents from different classes. A *pure* cluster contains only documents from only one ground truth cluster. Cluster contamination measure of a pure cluster is 0. In contrast, if a cluster contains documents from more than one partition, it is contaminated within 0-1 range. Different than other measures, lower contamination means better clustering.

- **Purity:** Cluster purity, on the other hand, expose the purity of a cluster, by assigning it to the class most frequent in it, and summing up number of common documents and dividing by number of documents gives us the purity of algorithm generated clusters. Disadvantage of purity is that it increases as number of clusters increase. More information can be found in [28].

- **Normalized mutual information (NMI):** It measures the amount of information we obtain about classes if we know the clusters. It computes and sums up mutual information between each class and cluster to calculate the overall success. Afterwards, normalization is obtained by considering the entropy of clusters and classes. More information can be found in [28]. Due to the *normalization* of mutual information, it can be used for evaluating methods with varying number of clusters.

We use the implementations of $w_{\text{F-measure}}$, contamination and normalized mutual information directly from Carrot^2 API [42]. This approach allows the comparison of the proposed method's result with the methods that will be developed in the future, using the same datasets and metrics.

4.2 Labeling Evaluation

In most of the SRC methods human judgment is preferred to evaluate the labeling performance; however, this approach is very expensive and difficult to repeat for different methods or parameters. Due to such drawbacks, it is difficult to compare distinct labeling methods based on human judgment. So, we propose a new metric called $\text{sim}_{\text{F-measure}}$ to automatically evaluate the success of generated labels.

4.3 Comparison of Ground Truth and Generated Label

The measure $\text{sim}_{\text{F-measure}}$ is based on the assessment of similarity between two labels (ground truth and generated label). We use four *similarity metrics* to automatically find similarity between generated label and ground truth label and they are semantic similarity, exact, partial and overlap match. Each similarity metric reflects the labeling performance from different aspects. While exact match is strict to the ground truth, partial match requires the ground truth structure (order of words) is preserved partially. Overlap match considers how close suggested labels are to the ground truth. Lastly, semantic similarity finds the underlying semantic relationship between labels.

Semantic similarity. Semantic similarity is a research field in artificial intelligence, that aims to determine the similarity between concepts by mapping them into an ontology and investigating their relationship within the ontology. In this thesis, we use semantic similarity to detect the similarity between the ground truth and proposed label. For the experiments, we use Java WordNet Similarity Library [32] that exploits WordNet [29] as the ontology source.

The semantic similarity metric outputs a similarity value within the range of 0 and 1 to quantify the similarity between two labels. Although there are different formulations of this metric, we are using the algorithm presented in [22] that works based on the formulation of [34] by using the information content concept of information theory. For example, in our experiments, ground truth and generated label pairs “News” - “Broadcasts” and “Sound Files” - “Streaming Audio” are found to share respectively 0.90 and 0.78 similarity according to the semantic similarity metric of [22].

During label evaluation, if the ground truth class is *Others* cluster, and algorithm cluster is not, or vice versa, the similarity score between labels is set to 0. In addition to the semantic similarity metric, we also use partial, exact and overlap match metrics to evaluate the similarities between ground truth and proposed

labels. Unlike semantic similarity, they give Boolean output; 1 for similarity and 0 for dissimilarity. Before applying these metrics, stopwords are eliminated and stemming is applied to labels.

Exact match. It suggests similarity if the generated label is the same as the ground truth or the generated label covers the other. To exemplify, when ground truth and generated label pair is "Instruments" - "Musical Instrument", exact match is ensured.

Partial match. It suggests similarity if the cluster label covers the ground truth label or vice versa. For instance, the ground truth - extracted label pair "USS Coral Sea, disambiguation" - "USS Coral Sea" is accepted, where the label found is the subset of the suggested label. The partial and exact match do not cover the case when the words in ground truth change order in generated label.

Overlap match. It aims to catch the slightest similarity between labels. If the intersection between the label and ground truth label is not empty, then the overlap match accepts the label. As an example; if the ground truth label is "Editorial Illustration", the overlap match accepts the generated label "Digital illustrations". Overlap match unlike the exact and partial match strategies, do not require that the order of words in ground truth are preserved in generated label.

4.4 Labeling Evaluation Measure: $\text{sim}_{\text{F-measure}}$

We decide to assess similarity between two labels by using similarity metrics, exact, partial, overlap match, and semantic similarity. In order to obtain a robust labeling evaluation metric for the entire clustering structure, we introduce a new measure, $\text{sim}_{\text{F-measure}}$, based on precision and recall. In this formulation, similarity precision ($\text{sim}_{\text{precision}}$) represents to what extend labels presented to the user resemble the ground truth labels and similarity recall ($\text{sim}_{\text{recall}}$) defines to what extend ground truth labels are reflected to the user.

The methodology for computing the overall similarity can be summarized as follows. For each class in the ground truth set, we find the matching cluster by using the highest F-measure principle. Then, we compute the similarity between the labels by using one of the similarity metrics (represented as *similarity* function in Equations 4.5 and 4.6). After that, we sum up the similarity scores for all classes and normalize by the number of classes to find the $\text{sim}_{\text{recall}}$. We find the $\text{sim}_{\text{precision}}$ by applying the same method for the clusters. Finally, the $\text{sim}_{\text{F-measure}}$ is computed as the harmonic mean of $\text{sim}_{\text{recall}}$ and $\text{sim}_{\text{precision}}$. This labeling measure is stable with respect to number of clusters, because it depends on both class and cluster perspectives coming from the harmonic mean. The necessary formulation for this procedure can be derived as follows.

$$sim_i = similarity\ \{label(class_i),\ label(cluster_{\max F\text{-}measure(i,j)})\} \quad (4.5)$$

$$sim_j = similarity\ \{label(cluster_j),\ label(class_{\max F\text{-}measure(i,j)})\} \quad (4.6)$$

$$sim_{precision} = \frac{\sum_{j=1}^{n_{cluster}} sim_j}{n_{cluster}} \quad (4.7)$$

$$sim_{recall} = \frac{\sum_{i=1}^{n_{class}} sim_i}{n_{class}} \quad (4.8)$$

$$sim_{F\text{-}measure} = \frac{2 \times sim_{recall} \times sim_{precision}}{sim_{recall} + sim_{precision}} \quad (4.9)$$

Equations 4.5 and 4.6 are described in Figure 4.3. Specifically, matching classes and clusters is shown for the calculation of similarity. To compute Equation 4.6, the arrows from each class to a cluster is drawn for the cluster with highest F-measure. To compute Equation 4.6, the dashed arrows from each cluster to a class is drawn for the class which gives the highest F-measure. Afterwards, the similarity between matched classes and clusters are calculated using similarity metrics; semantic similarity, exact, partial and overlap match. This step is represented as *similarity function* in Equations 4.5 and 4.6.

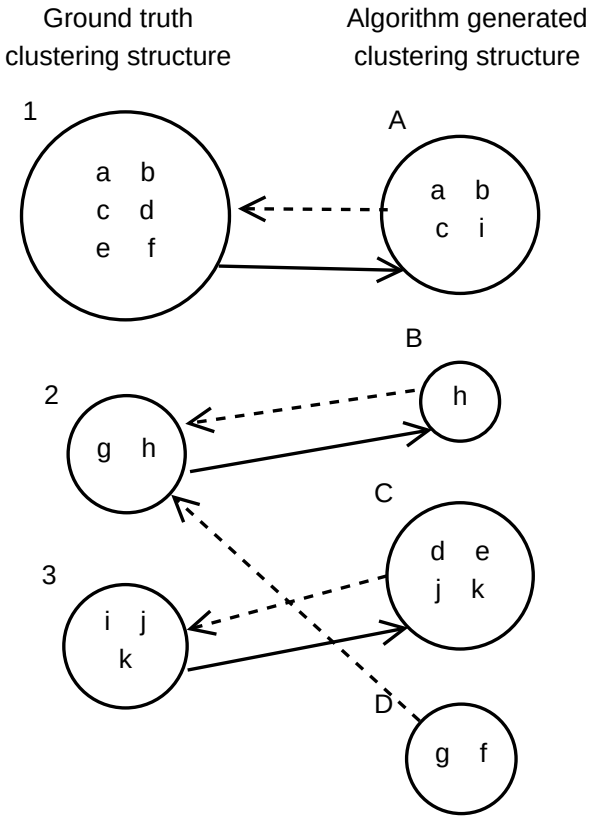

Figure 4.3: Demonstration of matching classes and clusters (that give maximum F-measure) for the calculation of similarity using Equations 4.5 and 4.6 for example clustering structures. Each class is matched with a cluster, that gives the highest F-measure among all clusters. The same operation is done for clusters too. We use each class and cluster match to compute the similarity between labels of them.

Chapter 5

Experimental Environment and Results

5.1 Experimental Environment

In order to assess the performance of the proposed algorithm, C^3M+K-means, we perform experiments in two publicly available datasets specific to SRC task: the Ambient Dataset [12] and ODP-239 Dataset [13]. The Ambient Dataset is constructed from ambiguous Wikipedia entries [3] and designed for testing with the ambiguous words that can be used for different meanings (subtopics). It includes 44 ambiguous queries and 100 snippets for each query that are obtained from a search engine. Each result in the dataset is represented with a URL, title and a very short text summarizing the web page. Human judgment is used for selecting the subtopic of each result from the meanings of the query given in Wikipedia. Note that the Ambient Dataset also includes some results that are not matched to any subtopic. To build a consistent evaluation scheme with our clustering method, we create *Others* cluster for these unmatched results and add this cluster to the ground truth.

The ODP-239 dataset consists of 239 queries, each with 100 snippets and about 10 subtopics. Each search result consists of a URL, title and a very short

Table 5.1: Statistical information about the Ambient and ODP-239 dataset

Property	Ambient	ODP-239
No. of query	44	239
Avg. No. of snippets per query	100	107.03
Total no. of snippets	4400	25580
Avg. No. of clusters per query	8.93	9.56
Avg. No. of snippets per cluster	12.45	11.38
Avg. No. of words per ground truth label	8.6	1.63
Avg. No. of words per snippet	30.02	19

text. The dataset is derived from Open Directory Project (ODP) [2], so human judgment is not needed for obtaining ground truth clustering. Unlike the Ambient dataset, no result is left unclustered in the ODP-239 dataset. Ambient focuses on search result clustering task with results that belong to different subtopics of an ambiguous query. Additionally, the results are retrieved from a search engine. Whereas ODP-239 focuses on multi-topic queries and classification task with results that belong to different categories of ODP.

Table 5.1 shows statistical properties of two datasets. It demonstrates the differences between datasets. In Section 5.2, the affect of datasets on experimental results is discussed from the point of view of the statistical information provided in this table.

5.2 Experimental Results

This section provides the evaluation results of our approach to clustering and labeling tasks in the datasets described in Section 5.1. We present both the results of C^3M and C^3M+K-means methods to discuss the effect of using sequential k-means clustering. We also provide the success of suffix tree clustering and Lingo in the datasets to compare the performance our algorithm with these prominent methods.

5.2.1 Clustering Results

In this part, we aim to measure the success of the generated partitioning (flat) clusterings of methods. First step is to prove that the proposed method performs statistically different than random clustering. Afterwards, we address the performance of the proposed method using mainly $w_{\text{F-measure}}$ measure. We also provide supportive measurements for clustering evaluation.

5.2.1.1 Clustering Performance versus Random Clustering

The first step of the clustering evaluation is to prove that the algorithm shows significant difference from random clustering according to the Monte Carlo method [21] as described in Section 4.1.1. Figures 5.1 and 5.2 present the histograms of the first queries random clusterings average target clusters for 1000 random clusterings and the proposed method's average number of target cluster (n_t) is shown with dashed line. According to these plots, the proposed algorithm's performance is better than all of the random clusterings in each dataset. We use *Monte Carlo %* term, to represent the percentage of random clusterings outperformed by the proposed method, which is %100 in the figures. For all queries, on the average, the proposed method outperforms %97.3 (in Ambient) and %98.8 (in ODP-239) of the 1000 random clusterings. In Table 5.2, comparison of the random clusterings with respect to the proposed method are given (average of n_{tr} and n_t values for all queries). It is observed that, the proposed method performs significantly different from random with smaller number of target clusters than random.

We apply two-tailed and paired t-test to prove the significant difference for each query as described in Section 4.1.1. For each query we obtain two distributions, for the proposed method and average of random clusterings (from n_{tr} and n_t samples of all queries). We use t-test to compare two distributions with each other. The data for this test is presented in Appendix. In our experiments, we prove the statistical significance of the proposed method with respect to random clusterings using t-test, as in the Monte Experiment.

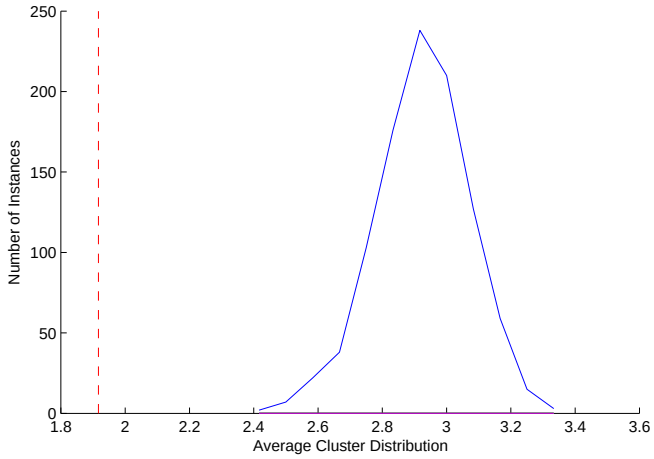

Figure 5.1: The Monte Carlo experiment for the first query of the Ambient dataset. Histogram of 1000 randomly generated clusterings' average number of target clusters with respect to the proposed method's n_t value (shown as a dashed line)

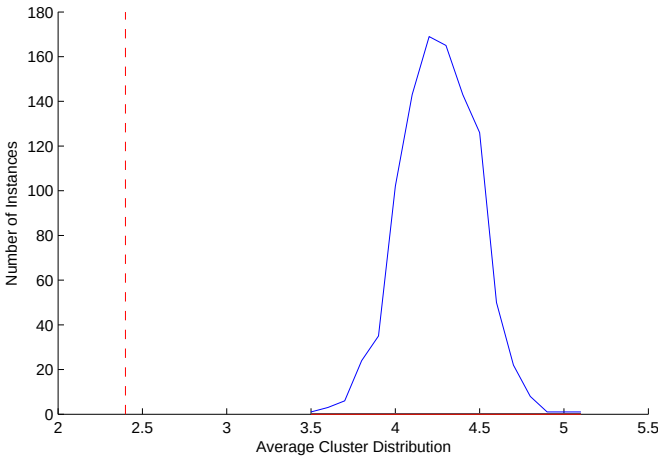

Figure 5.2: The Monte Carlo experiment for the first query of the ODP-239 dataset. Histogram of 1000 randomly generated clusterings' average number of target clusters with respect to the proposed method's n_t value (shown as a dashed line)

Table 5.2: Clustering analysis with random clustering in terms of n_t, n_{tr} and Monte Carlo % metrics.

Dataset	Algorithm	n_t	n_{tr}	Monte Carlo %
Ambient	C^3M	3.3903	3.9895	91.1409
	C^3M+K-means	2.4698	3.6064	97.275
ODP-239	C^3M	4.2195	5.0446	95.8393
	C^3M+K-means	3.2096	4.6262	98.7619

Table 5.3: Clustering results in terms of $w_{\text{F-measure}}$

Dataset	Algorithm	$w_{\text{F-measure}}$
Ambient	C^3M	0.444
	C^3M+K-means	**0.603**
	STC	0.413
	Lingo	0.370
ODP-239	C^3M	0.386
	C^3M+K-means	0.464
	STC	**0.510**
	Lingo	0.420

5.2.1.2 Clustering Results with Main Evaluation Measure: $w_{\text{F-measure}}$

In this section, we demonstrate the clustering performance of the proposed method in a comparative manner. We test our algorithm in the same datasets by using main evaluation measure $w_{\text{F-measure}}$ success measure. Table 5.3 details the average results for all queries in both datasets including the results for suffix tree clustering and Lingo algorithms.

As seen in Table 5.3, the proposed C^3M+K-means method performs the best among all methods in the Ambient dataset when we look at the $w_{\text{F-measure}}$ results. To prove the statistically significance of our results with respect to the other algorithms, we also run a two-tailed, paired t-test over $w_{\text{F-measure}}$ scores of all queries in Ambient. With a threshold level of 0.01, we achieve statistical significance in our results with very small p-values ($1.28\text{x}10^{-9}$ and $2.53\text{x}10^{-13}$ for STC and

Lingo respectively).

On the other hand, the proposed method ranks second in the ODP-239 dataset with a %4.6 difference. The experiments show that this difference between STC and C^{3}M+K-means is not statistically significant, so the proposed algorithm is also successful in clustering the ODP-239 dataset. Notice that, the usage of sequential k-means as a secondary clustering mechanism after the C^{3}M method increases the clustering performance significantly. Obtaining high success rates in Ambient is more important since it contains ambiguous queries with results derived from a search engine, is more similar to real life SRC tasks, while ODP-239 is more suitable for classification tasks.

Now, we criticize our approach for clustering by considering alternative methods. In order to discuss the importance of C^{3}M algorithm in the proposed method, we perform *randomly started k-means* (random k-means). For the stabilization of random behavior, each query of the datasets is run for 1000 times and the average of $w_{\text{F-measure}}$ scores is set as the result of the query. The results are shown in Table 5.2.1.2. If we repeat again, random k-means gives on average 0.526 and 0.437 respectively in Ambient and ODP-239. Moreover, we perform the *original k-means* instead of sequential k-means after C^{3}M algorithm, resulting 0.481 and 0.457 $w_{\text{F-measure}}$ scores in two datasets. If we compare these outputs with respect to the proposed method C^{3}M+K-means' success, we observe that the difference among clustering methods are more obvious in Ambient than the other dataset. We show that using both C^{3}M and sequential k-means is important because this association increases the success of clustering in both datasets, especially in the Ambient dataset.

5.2.1.3 Clustering Results with Supportive Measures

Additionally, we present clustering results for three more metrics: Purity, contamination and NMI in Table 5.5. We discuss important features of these metrics in Section 4.1. Purity and contamination are reflect counterparts of the same concept *pureness* of clustering. NMI measures the success of clustering from an

Table 5.4: Table shows the $w_{\text{F-measure}}$ scores of alternative methods and proposed method C^{3}M+K-means

Method	Ambient	ODP-239
Random K-means	0.526	0.437
C^{3}M+Original K-means	0.481	0.457
C^{3}M+K-means	0.603	0.464

Table 5.5: Clustering results in terms of purity, contamination and NMI metrics

Dataset	Algorithm	Purity	Contamination	NMI
Ambient	C^{3}M	0.628	0.588	0.262
	C^{3}M+K-means	0.738	0.440	0.419
	STC	0.768	0.380	**0.459**
	Lingo	**0.818**	**0.332**	0.452
ODP-239	C^{3}M	0.507	0.706	0.305
	C^{3}M+K-means	0.580	0.605	0.392
	STC	0.5941	0.583	0.424
	Lingo	**0.711**	**0.470**	**0.476**

information theoretic perspective. As seen in Table 5.5, Lingo shows better performance with respect to these metrics except the higher effectiveness of STC with NMI evaluation metric in Ambient dataset.

Experimental results for clustering are presented in this Section. If we discuss affect of the datasets on the success results obtained, we can consult to Table 5.1. It demonstrates the statistical differences between datasets. For clustering measures, methods success in the ODP-239 dataset is lower with respect to Ambient dataset, except for the NMI metric. The gap between them is especially obvious for $w_{\text{F-measure}}$, which is the main metric we consult for clustering evaluation. The reason behind this is possibly resulting from the average number of words per snippet property, Ambient and ODP-239 contains 30 and 19 words per snippet respectively in Table 5.1. We can conclude that lesser information inherent in ODP-239 makes the collection harder to cluster.

Table 5.6: Ground truth labels and matching labels generated by the proposed method for the "Water Sports" query of ODP-239. Exact (E), partial (P), overlap (O) match and semantic (S) similarity scores are provided.

Ground Truth Label	Generated Label	E	P	O	S
Swimming and Diving	United States	0	0	0	0.40
Canoeing and Kayaking	Photo	0	0	0	0.42
Rowing	Row	1	1	1	1.00
Surfing	Surf	1	1	1	1.00
Water Skiing and Wakeboarding	Wakeboarding	0	1	1	1.00
Water Polo	Water Polo	1	1	1	1.00
Dragon Boating	Dragon Boat	1	1	1	1.00
Windsurfing	Windsurfing	1	1	1	1.00
Kitesurfing	Photo	0	0	0	0.00
Surf Life Saving	Surf	0	1	1	1.00
-	Pictures	-	-	-	-
-	Build	-	-	-	-

5.2.2 Labeling Results

This section discusses the experiments we performed to evaluate the proposed labeling method. In Table 5.6, example labels of the proposed method are shown for the "Water Sports" topic of the ODP-239 dataset. Our method can successfully determine some of the labels as seen in the table. As seen in this example "Rowing" and "Row" are determined as similar, due to the stemming performed before the labeling evaluation as described in Section 4.2 (except the semantic similarity, which handles this step itself). However, sometimes it sets insignificant terms as labels. Unfortunately, semantic similarity metric find likeness between unrelated labels in the first two rows using the WordNet hierarchy.

From now on we will inspect Table 5.7 which details the labeling performances. Success rates in the Ambient and ODP-239 datasets are shown based on the semantic similarity, exact, partial, and overlap match metrics applied on *similarity F-measure* ($\text{sim}_{\text{F-measure}}$) label evaluation measure. In contrast to the smaller exact match scores by all methods in Ambient relative to ODP-239, we observe higher scores in the other measures. The reason behind is that the ground

Table 5.7: Labeling results in terms of $\text{sim}_{\text{F-measure}}$. Similarity between labels are decided by Exact (E), Partial (P), Overlap (O) match and semantic similarity (S) metrics.

Dataset	Algorithm	E	P	O	S
Ambient	C^3M	0.002	0.151	0.481	0.214
	C^3M+K-means	0.005	0.235	**0.488**	0.261
	STC	**0.086**	**0.335**	0.455	**0.331**
	Lingo	0.049	0.209	0.406	0.225
ODP	C^3M	0.091	0.112	0.149	0.108
	C^3M+K-means	**0.151**	**0.185**	**0.221**	**0.172**
	STC	0.119	0.176	0.195	**0.172**
	Lingo	0.112	0.144	0.168	0.137

truth labels, which define the meaning of ambiguous words, are too long in the Ambient dataset. For example, "Aida, the musical by Elton John and Tim Rice whose story is based on that of the opera" is one of the subtopics of query "Aida". On the other hand, in general, ODP-239 contains one to three-word subtopics, e.g., "News and Media". Note that scores are low by all methods. Due to the labeling evaluation strategy, success of labeling depends on how good clusters are obtained.

For the Ambient dataset, our algorithm performs best in overlap match, while ranking second in other measures following the STC algorithm except the exact match (in exact match case Lingo is the second best). We show the significance of these results using a t-test as described previously. In contrast, our method outperforms the other methods in all the success metrics in the ODP-239 dataset except the semantic similarity match (in semantic similarity match case it is in tie with STC). However, statistical significance is not observed due to the close results of the proposed method and STC. In the light of these results, it can be concluded that, the proposed method shows comparative performance on labeling clusters.

In fact, the automatically computed similarity metrics are more strict than

human judgment and they produce smaller similarity scores since they only compare with ground truth label, while human can also consider cluster content. In addition, automatic evaluation finds the similarity between labels if they share words or have a relationship in the ontology, but human infers similarity intuitively, even such an association do not exists. However, the disadvantage of such an evaluation method is that the results may vary from person to person. Therefore, we can say that, using an automatic similarity metric simplifies the comparison of search result labeling methods. Inserting F-measure into the computation of $\text{sim}_{\text{F-measure}}$ provides that the cluster content should match with the class content. This ensures that not only the label similarity is enough but also the documents in the cluster should be common with the ground truth subtopic.

Table 5.1 shows statistical properties of two datasets. From the labeling point of view, as we state above, the ODP-239 dataset labels are much more shorter than Ambient. Average number of words per ground truth label property is 8.6 and 1.63 for Ambient and ODP-239 respectively. The success of labeling is lower in ODP-239 dataset, which is calculated by measuring the similarity between labels. Therefore, providing more information about the label (in Ambient dataset), may help the evaluation of labeling. Also, longer descriptions decrease the strictness of ground truth labels, by allowing different combinations of generated labels.

5.2.3 Time Performance

Lastly, we measure the average process time for one query for the two datasets. Results are shown in Table 5.8, where time is presented in milliseconds. The methods are run in a computer with properties described below:

- Processor: Pentium M 740 Centrino - 1.73 GHz 533 Mhz SpeedStep (2 MB Cache)
- RAM: 1 GB DDR 333-SDRAM (Max.1 GB)
- Operating system: Ubuntu 10.10

Table 5.8: Average time performances in terms of millisecond

Dataset	Algorithm	Average Time Performance
Ambient	C^3M+K-means	390.20
	STC	32.18
	Lingo	307.30
ODP	C^3M+K-means	300.34
	STC	18.03
	Lingo	271.50

While STC is very efficient in terms of time with respect to the other two methods, the proposed method and Lingo. The proposed method ranks the third in terms of time performance, but the values for Ambient and Lingo shows that, our method is still appropriate for search result clustering by presenting results in 0.3 and 0.4 seconds for the two datasets. In addition, our method's performance is comparable with Lingo's performance.

Chapter 6

Conclusion

In this thesis, we propose an approach for solving a key information retrieval problem: search result clustering which covers both clustering and labeling of search results. Our study addresses the difficulty of this problem with motivation and necessity of it. Our main contribution on SRC can be summarized as taking document relationships into account by using cover coefficient clustering method and using its results as an initial clustering structure for the well-known sequential k-means clustering algorithm for improving the SRC performance. Another contribution of this study is a novel cluster labeling approach called "labeling via term weighting." The method observes both the behavior of terms within the cluster and in the entire search results for the query.

Our approach for the evaluation of SRC is also a contribution of this thesis. We introduce a new metric to assess the effectiveness of cluster labeling, namely similarity F-measure ($\text{sim}_{\text{F-measure}}$). This measure employs traditional precision and recall metrics. The resemblance between the generated and ground truth labels is determined by exact, partial, and overlap match similarity metrics that we introduce in this thesis. Also use of semantic similarity research area is suggested for resemblance determination, during labeling evaluation.

Extensive experimental results for both clustering and labeling show that the proposed method successfully cluster and label search results. It also maintains a

performance competitive with the two state-of-the-art methods used for clustering and cluster labeling; namely Lingo and suffix tree clustering. The experiments run on publicly available two datasets, Ambient and ODP-239, specially created for SRC task.

Future pointers for enhancing SRC problem are:

1. A new approach for estimating number of clusters can be utilized, instead of setting number of clusters to a fixed size.

2. A new method for converting flat (partitioning) clustering structures to overlapping can be utilized.

3. Creation of a Turkish dataset enables the application of search result clustering problem to Turkish. This dataset is useful for solving problems originating from the structure of Turkish language. Especially, the success of phrase extraction phase may not be observed for Turkish language. Measuring the success of suffix tree based phrase extraction for Turkish can be valuable. Hence, proposing new solutions for finding phrases in Turkish language in limited amount of time can be a future pointer.

4. The success of SRC problem highly depends on the features extracted from snippets. The features that reveal the differences between snippets are valuable for clustering. And features that are meaningful, expressive and natural to human are valuable for labeling. Therefore, utilizing new feature extraction methods for extracting features for clustering and labeling tasks could increase the success of SRC methods.

5. An evaluation for the evaluation measures' stability similar to [5] can be valuable. By means of this study, standard evaluation metrics for SRC can be proposed. The approach for dealing with "Others" cluster used in SRC clustering structure heavily affects the measurement results.

Bibliography

[1] Google search engine. `http://www.google.com.tr`. Accessed on August, 19, 2011.

[2] Open directory project. `http://www.dmoz.org/`. Accessed on August, 19, 2011.

[3] Wikipedia: Links to (disambiguation) pages. `http://en.wikipedia.org/wiki/Wikipedia:Disambiguation_pages_with_links`. Accessed on August, 19, 2011.

[4] R. B. Allen, P. Obry, and M. L. Littman. An interface for navigating clustered document sets returned by queries. In *COOCS*, pages 166–171. ACM, 1993.

[5] C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR) Conference on Research and Development in Information Retrieval*, pages 33–40, 2000.

[6] F. Can, I. S. Altingovde, and E. Demir. Efficiency and effectiveness of query processing in cluster-based retrieval. *Information Systems*, 29(8):697–717, 2004.

[7] F. Can, S. Kocberber, O. Baglioglu, S. Kardas, H. C. Ocalan, and E. Uyar. Bilkent news portal: a personalizable system with new event detection and tracking capabilities. In S.-H. Myaeng, D. W. Oard, F. Sebastiani, T.-S. Chua, and M.-K. Leong, editors, *Proceedings of the Special Interest Group*

on Information Retrieval (SIGIR) Conference on Research and Development in Information Retrieval, page 885. ACM, 2008.

[8] F. Can and E. A. Ozkarahan. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *ACM Transactions on Database Systems*, 15(4):483–517, 1990.

[9] J. G. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Research and Development in Information Retrieval*, pages 335–336, 1998.

[10] C. Carpineto, S. Mizzaro, G. Romano, and M. Snidero. Mobile information retrieval with search results clustering: Prototypes and evaluations. *Journal of the American Society for Information Science and Technology*, 60(5):877–895, 2009.

[11] C. Carpineto, S. Osinski, G. Romano, and D. Weiss. A survey of web clustering engines. *ACM Computing Surveys*, 41(3), 2009.

[12] C. Carpineto and G. Romano. Ambient dataset. `http://credo.fub.it/ambient/`, 2008. Accessed on August, 19, 2011.

[13] C. Carpineto and G. Romano. Odp-239 dataset. `http://credo.fub.it/odp239/`, 2009. Accessed on August, 19, 2011.

[14] W. B. Croft. *Organizing and Searching Large Files of Documents*. PhD thesis, University of Cambridge, 1978.

[15] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg. Alignment of whole genomes, 1999.

[16] S. T. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 23(2):229–236, 1991.

[17] E. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–780, 1965.

[18] D. Gusfield. *Algorithms on Strings, Trees, and Sequences : Computer Science and Computational Biology*. Cambridge Univ. Press, January 2007.

[19] P. Hannappel, R. Klapsing, A. Krug, and G. Neumann. MSEEC - a multi search engine with multiple clustering. In M. KhosrowPour, editor, *Managing Information Technology Resources in Organizations in the Next Millennium: Proceedings of the 10th Information Resources Management Association International Conference*. Idea Group Publishing, 1999.

[20] M. Hearst and J. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR) Conference on Research and Development in Information Retrieval*, pages 76–84. ACM, 1996.

[21] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[22] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference Research on Computational Linguistics (ROCLING)*, Taiwan, 1997.

[23] Y. Kural, S. E. Robertson, and S. Jones. Deciphering cluster representations. *Information Processing and Management*, 37(4):593–601, 2001.

[24] Landauer. *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum Associates, 2007.

[25] A. Leouski and W. Croft. An evaluation of techniques for clustering search results. 1996.

[26] L. Lim, M. Wang, S. Padmanabhan, J. S. Vitter, and R. Agarwal. Dynamic maintenance of web indexes using landmarks. In *Proceedings of the Twelfth International World Wide Web Conference*, pages 102–111, Budapest, Hungary, May 2003. ACM Press.

[27] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. L. Cam and J. Neyman, editors, *Proceedings of the fifth*

Berkeley Symposium on Mathematical Statistics and Probability, volume 1, pages 281–297. University of California Press, 1967.

[28] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[29] G. A. Miller and C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[30] H. C. Ocalan. *Bilkent News Portal: A System with New Event Detection and Tracking Capabilities*. PhD thesis, Bilkent University, Ankara, Turkey, 2009.

[31] S. Osinski, J. Stefanowski, and D. Weiss. Lingo: Search results clustering algorithm based on singular value decomposition. In *Intelligent Information Systems*, pages 359–368, 2004.

[32] G. Pirro. A semantic similarity metric combining features and intrinsic information content. *Data and Knowledge Engineering*, 68(11):1289–1308, 2009.

[33] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[34] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (AAAI)*, pages 448–453, 1995.

[35] J. J. Rocchio. Document retrieval systems - optimization and evaluation. 1966.

[36] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[37] N. Seco, T. Veale, and J. Hayes. An intrinsic information content metric for semantic similarity in WordNet. In *European Conference on Artificial Intelligence*, pages 1089 – 1090. IOS Press, 2004.

[38] A. Turel and F. Can. A new approach to search result clustering and labeling. In *submitted*, 2011.

[39] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[40] P. Weiner. Linear pattern matching algorithms. In *The Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–11. IEEE, 1973.

[41] D. Weiss. *Descriptive Clustering as a Method for Exploring Text Collections*. PhD thesis, Poznan University of Technology, Poznan, Poland, 2006.

[42] D. Weiss and S. Osinski. Carrot^2 open source search results clustering engine. `http://project.carrot2.org/`, 2002. Accessed on August, 19, 2011.

[43] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR) Conference on Research and Development in Information Retrieval*, 1998.

[44] H. J. Zeng, Q. C. He, Z. Chen, W. Y. Ma, and J. Ma. Learning to cluster web search results. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR) Conference on Research and Development in Information Retrieval*, pages 210–217, Sheffield, United Kingdom, 2004. ACM Press.

[45] D. Zhang. *Towards Web Information Clustering*. PhD thesis, Southeast University, Nanjing, China, 2002.

[46] D. Zhang and Y. Dong. Semantic, hierarchical, online clustering of web search results. In J. X. Yu, X. Lin, H. Lu, and Y. Zhang, editors, *Asia-Pacific Web Conference*, volume 3007 of *Lecture Notes in Computer Science*, pages 69–78. Springer, 2004.

[47] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2):6, 2006.

Appendix A

Data

Table A.1 and Table A.2 present the data used in Section 5.2.1.1 for Ambient and ODP-239 datasets. The data given is used during t-test experiments to prove the statistical significance of the proposed method. More information about these experiments can be found in Sections sec:random:evaluation and sec:cluster:results:random.

Table A.1: Comparison of the proposed method's output and random clustering with respect to average number of target clusters for each query in Ambient dataset is presented. n_{tr} is the random clustering's and n_t is the proposed method's abbreviation for average number of target clusters. n_{tr} is obtained from the average of *average number of target clusters* of 1000 random clusterings.

Query No	$\mathbf{n}_t$	$\mathbf{n}_{tr}$	**Query No**	$\mathbf{n}_t$	$\mathbf{n}_{tr}$
1	1.92	2.92	23	2.33	3.61
2	4.25	7.29	24	2.63	4.09
3	3.40	5.14	25	2.50	3.66
4	3.80	5.49	26	1.93	2.39
5	2.44	3.01	27	2.00	3.01
6	2.86	4.02	28	3.50	5.66
7	3.80	5.72	29	2.10	3.68
8	2.00	3.38	30	2.33	2.61
9	2.11	3.84	31	2.75	4.02
10	2.22	3.01	32	2.11	2.82
11	2.33	4.37	33	3.00	5.49
12	1.92	3.29	34	2.71	4.10
13	2.30	3.29	35	1.77	2.26
14	1.75	2.29	36	3.25	3.65
15	3.25	4.74	37	2.43	3.47
16	2.71	4.36	38	3.00	3.87
17	2.63	4.59	39	2.64	3.36
18	2.43	3.34	40	1.87	2.47
19	1.69	2.29	41	2.00	3.39
20	4.00	4.47	42	2.43	3.48
21	2.88	4.41	43	2.38	3.02
22	2.50	3.76	44	2.09	2.82
			Average	2.57	3.77

Table A.2: Comparison of the proposed method's output and random clustering with respect to average number of target clusters for each query in ODP-239 dataset is presented. n_{tr} is the random clustering's and n_t is the proposed method's abbreviation for average number of target clusters. n_{tr} is obtained from the average of *average number of target clusters* of 1000 random clusterings.

Query No	$\mathbf{n}_t$	$\mathbf{n}_{tr}$	**Query No**	$\mathbf{n}_t$	$\mathbf{n}_{tr}$	**Query No**	$\mathbf{n}_t$	$\mathbf{n}_{tr}$
1	2.40	4.26	31	2.40	4.54	61	2.90	4.37
2	3.10	4.39	32	3.20	4.73	62	3.20	4.19
3	3.50	4.00	33	4.30	4.45	63	4.33	4.12
4	3.00	4.14	34	3.20	4.72	64	2.78	4.51
5	2.60	5.36	35	4.20	5.19	65	3.30	4.04
6	3.29	4.46	36	3.30	4.97	66	4.17	6.03
7	3.80	5.13	37	3.30	4.88	67	2.70	3.54
8	3.80	4.69	38	3.10	4.69	68	3.20	4.46
9	2.80	3.81	39	3.80	5.27	69	3.40	5.07
10	2.70	5.08	40	3.30	4.56	70	3.40	4.76
11	3.78	5.41	41	2.90	4.48	71	3.40	4.57
12	4.60	5.31	42	2.90	4.39	72	3.30	5.15
13	2.50	4.74	43	2.90	5.13	73	3.50	4.99
14	3.00	4.08	44	2.50	4.43	74	3.67	5.21
15	2.90	4.12	45	2.80	4.69	75	3.40	4.07
16	2.90	3.96	46	4.10	4.84	76	2.80	5.12
17	4.57	5.32	47	3.20	4.43	77	4.00	4.97
18	3.00	5.18	48	2.90	4.63	78	3.00	3.58
19	3.40	4.57	49	2.89	4.84	79	2.80	4.18
20	2.50	3.87	50	3.90	5.35	80	3.70	4.89
21	2.20	4.54	51	3.40	5.07	81	3.10	6.01
22	3.40	4.53	52	3.20	4.64	82	3.00	3.92
23	2.70	5.01	53	3.00	4.20	83	3.38	5.09
24	3.30	4.65	54	3.00	4.29	84	3.20	4.20
25	2.90	4.32	55	4.40	5.28	85	4.00	5.44
26	2.90	4.36	56	3.00	5.21	86	3.30	4.19
27	3.60	4.63	57	3.10	4.63	87	4.10	5.27
28	3.14	5.96	58	3.90	5.20	88	3.10	4.54
29	4.20	5.57	59	3.50	5.82	89	3.40	4.35
30	3.80	5.20	60	3.20	5.42	90	4.40	4.72

Query No	$\mathbf{n}_t$	$\mathbf{n}_{tr}$	**Query No**	$\mathbf{n}_t$	$\mathbf{n}_{tr}$	**Query No**	$\mathbf{n}_t$	$\mathbf{n}_{tr}$
91	3.50	4.37	121	3.30	3.98	151	3.00	3.50
92	3.70	4.72	122	3.10	3.98	152	3.00	4.62
93	2.50	4.46	123	3.60	4.15	153	2.60	4.06
94	3.00	3.79	124	3.30	4.47	154	3.25	5.15
95	2.86	5.02	125	2.70	4.70	155	2.90	4.33
96	3.38	3.94	126	1.90	3.60	156	2.80	3.75
97	3.20	4.18	127	2.40	3.61	157	3.40	5.68
98	3.00	4.80	128	4.00	4.81	158	2.70	4.59
99	2.90	5.06	129	3.40	4.06	159	3.11	4.80
100	3.30	5.11	130	3.33	5.84	160	2.80	5.34
101	2.70	4.03	131	4.14	5.98	161	3.30	4.58
102	3.30	4.82	132	3.90	4.68	162	2.90	4.72
103	3.75	5.51	133	2.40	4.49	163	3.70	4.46
104	2.70	4.61	134	2.80	4.31	164	3.20	4.72
105	3.50	5.19	135	3.50	4.24	165	4.67	6.72
106	2.60	3.34	136	2.80	3.91	166	3.80	5.62
107	3.30	4.87	137	3.40	5.19	167	3.40	4.76
108	1.90	4.92	138	3.50	4.83	168	3.00	4.23
109	2.90	4.60	139	3.40	4.93	169	2.83	4.67
110	4.20	5.62	140	3.50	4.65	170	4.00	4.86
111	3.56	4.18	141	3.33	4.20	171	3.20	5.49
112	3.10	4.70	142	3.10	3.51	172	5.30	5.88
113	3.30	4.49	143	3.80	4.95	173	3.00	4.95
114	2.90	3.87	144	2.40	4.21	174	2.70	4.43
115	3.83	4.04	145	3.00	5.52	175	3.60	5.33
116	2.90	4.38	146	3.70	4.73	176	3.80	4.86
117	2.80	4.79	147	3.80	4.63	177	3.10	4.88
118	3.33	4.93	148	3.40	5.01	178	2.90	4.56
119	2.90	3.87	149	3.40	4.50	179	4.00	4.84
120	3.30	5.10	150	3.30	5.09	180	2.38	3.89

Query No	$\mathbf{n}_t$	$\mathbf{n}_{tr}$	**Query No**	$\mathbf{n}_t$	$\mathbf{n}_{tr}$
181	2.70	4.29	211	3.17	4.45
182	2.40	5.80	212	3.60	4.26
183	4.43	5.94	213	3.10	4.10
184	3.20	4.80	214	4.67	5.50
185	2.86	5.70	215	3.60	5.34
186	4.00	5.31	216	3.40	4.80
187	2.70	3.20	217	2.80	3.98
188	2.70	4.83	218	3.10	4.49
189	3.20	5.03	219	3.25	5.04
190	2.60	4.38	220	2.20	3.58
191	3.40	4.57	221	2.90	4.69
192	4.30	5.13	222	2.70	5.05
193	3.10	3.89	223	2.40	3.98
194	2.70	3.65	224	3.60	5.09
195	2.80	4.31	225	2.80	4.29
196	2.40	4.11	226	2.33	4.14
197	2.90	4.79	227	3.70	4.33
198	2.90	4.32	228	3.30	4.30
199	4.29	5.83	229	2.40	3.47
200	3.30	4.23	230	3.10	4.43
201	2.80	4.52	231	3.57	4.24
202	3.10	4.55	232	3.10	4.40
203	2.50	4.27	233	2.80	5.14
204	4.20	4.67	234	3.50	4.71
205	3.40	4.39	235	3.40	4.18
206	3.60	4.76	236	3.10	4.23
207	2.70	4.15	237	2.50	3.66
208	2.60	2.93	238	3.20	4.38
209	2.90	4.74	239	2.30	4.89
210	2.90	4.85			
			Average	3.22	4.65