NTNU
Norwegian University of
Science and Technology

Authoritative K-Means for Clustering of Web Search Results

Gaojie He

Master in Information Systems

Submission date:	June 2010
Supervisor:	Kjetil Nørvåg, IDI
Co-supervisor:	Robert Neumayer, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

In web search, surfers are often faced with the problem of selecting their most wanted information from the potential huge amount of search results. The clustering of web search results is the possible solution, but the traditional content based clustering is not sufficient since it ignores many unique features of web pages. The link structure, authority, quality, or trustfulness of search results can play even the higher role than the actual contents of the web pages in clustering. These possible extents are reflected by Google's PageRank algorithm, HITS algorithm and etc. The main goal of this project is to integrate the authoritative information such as PageRank, link structure (e.g. in-links and out-links) into the K-Means clustering of web search results. The PageRank, in-links and out-links can be used to extend the vector representation of web pages, and the PageRank can also be considered in the initial centroids selection, or the web page with higher PageRank influences the centroid computation to a higher degree. The relevance of this modified K-Means clustering algorithm needs to be compared to the ones obtained by the content based K-Means clustering, and the effects of different authoritative information also needs to be analyzed.

Assignment given: 18. January 2010
Supervisor: Kjetil Nørvåg, IDI

Authoritative K-Means For Clustering of Web Search Results

Gaojie He

Supervisor:

Robert Neumayer
Kjetil Norvag

June 14, 2010

Abstract

Clustering is currently more and more applied on hyperlinked documents, especially for web search results. Although most commercial web search engines will provide their ranking algorithms sorting the matched results to raise the most relevant pages to the top, the size of results is still so huge that most ones including some pages that suffers are really interested in will be discarded. Clustering for web search results separates unrelated pages and clusters the similar pages with the same topic into the same group, thus helps suffers to locate the pages much faster. Many features of web pages have been studied to be used in clustering, such as content information including title, snippet, anchor text and etc. Hyperlink is another primary feature of web pages, some content-link coupled clustering methods have been studied. We propose an authoritative K-Means clustering method that combines content, in-link, out-link and PageRank. In this project, we adjust the construction of in-link and out-link vectors and introduce a new PageRank vector with two patterns, one is a single value representation of PageRank and the other is a 11-dimensional vector. We study the difference of these two types of PageRank in clustering, and compare the different clustering based on different web page representations, such as content-based, content-link coupled and etc. The effect of different elements of web page is also studied in our project. We apply the authoritative clustering for the web search results retrieved from Google search engine. Three experiments are conducted and different evaluation metrics are adopted to analyze the results.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides a short introduction to the topics of the project. The background and motivation of this project is presented first, followed by the problem description and project goals. Next, we describe our contributions to the problem in our project. Finally, we provide the outline of this report.

1.1 Motivation

As the information capacity of Internet is growing massively day by day, it becomes nearly impossible to find the expected information by surfing through the whole Internet randomly. Therefore, most surfers turn to web search engine consciously to find the relevant information quickly with their needs. The web search engine is designed to search for information on the World Wide Web, and the search results are usually presented in a list of results that are commonly called hits, which may consists of web pages, images, videos, information fragments and other types of files.

Web search engine provides an effective interface for surfers to search for the information, but it also raises an inevitable problem: how to offer surfers the most relevant information within a specific query topic? The web search engines, such as Google, Yahoo, Bing and etc, usually come out with a potentially huge amount of results (e.g. There are totally 641,000,000 matched results by query "food", searched in May of 2010), surfers may only look at the top 50 or top 20 even fewer results and all the rest are discarded. Although most web search engines use different searching and ranking algorithms to list the most relevant results first, some results that the surfer is actually most interested in will still be ignored. Take such a scenario as an example: a surfer would like to know about a native creature called leopard, then he types "leopard" in a web search engine as the query but gets many results related to the Apple Company in priority since “leopard” is one of its Mac Operating System, and even worse, the surfer may get some results related to the jean if there is such a jean brand called leopard.

The categorization of information is a possible solution to this problem and help surfers to find the information much faster and preciser. Still stay with the previous scenario, if all the search results by the query “leopard” can be categorized into some different groups, e.g. a group possibly named as “animal” that contains all search results really talking about the native creature leopard, a group possibly named as “computer science” that contains all search results related to the Apple Mac OS and a group possibly named as “fashion” that contains all search results including the jean information, the surfer can go deep into the group “animal” directly to look for the information he is most interested in.

There are basically two types of categorization techniques: static categorization and dynamic categorization. The static categorization contains such as document classification, Google open directory, Wikipedia categories and etc. Supervised is the main feature of static categorization: each new document needs to be labeled with a class beforehand and a classifier is built to assign documents to one of the classes. The representative of dynamic categorization is document clustering. Unlike static categorization, the dynamic categorization is unsupervised. The assignment of documents to the clusters is performed by the clustering algorithm, similarity measure and the number of clusters. Clustering denotes the partitioning of documents in different groups. The documents are supposed to be similar to each other within the same group and dissimilar to documents that belong to different groups. K-means is one of the simplest clustering algorithms: random documents are first chosen as cluster centroids, each document is then assigned to its closest cluster centroid. Afterwards, these centroids are adjusted to reflect their currently assigned documents. The assignment of documents is repeated until some specific requirement is met.

Document representation is the fundamental part of clustering algorithm. The content based clustering using content vector in vector space model might be efficient for full independent pure text documents, but is sufficient for other document formats. For example, the authoritative information of academic papers, such as author, publisher, conference, reference and etc, combined with pure content are used to improve the clustering performance and quality. For hypertext documents such as web pages, hyper links are their unique feature. Therefore, the hyper links or other authoritative information derived from hyper links (e.g. PageRank) are combined with pure content of web pages to improve the clustering for web documents. This is the biggest motivation of this project.

1.2 Problem Description

In web search, surfers are often faced with the problem of selecting their most wanted information from the potential huge amount of search results. The clustering of web search results is the possible solution, but the traditional content based clustering is not sufficient since it ignores many unique features of web pages. The link structure, authority, quality, or trustfulness of search results can play even the higher role than the actual contents of the web pages in clustering. These possible extents are reflected by Google's PageRank algorithm, HITS algorithm and etc. The main goal of this project is to integrate the authoritative information such as PageRank, link structure (e.g. in-links and out-links) into the K-Means clustering of web search results. The PageRank, in-links and out-links can be used to extend the vector representation of web pages, and the PageRank can also be considered in the initial centroids selection, or the web page with higher PageRank influences the centroid computation to a higher degree. The relevance of this modified K-Means clustering algorithm needs to be compared to the ones obtained by the content based K-Means clustering, and the effects of different authoritative information also needs to be analyzed.

1.3 Our Contribution

In this project, we propose an authoritative K-Means clustering method for web search results, which combines the authoritative information of web page, such as Google PageRank, to the content-link coupled K-Means clustering. The representation of web page in content-link coupled clustering is generally composed of triple vector of content, in-links and out-links. By involving PageRank, the web page is represented as an quaternary vector including PageRank vector besides the previous triple vector. Our authoritative K-Means clustering method is based on the construction of this quaternary vector representation of web page. Specifically, our main contributions in this project can be summarized as:

- Propose a new vector construction pattern of in-links and out-links vectors. Unlike some researches take completed URL of web page as the item of link vector, we adopt the domain of URL as the vector item. The advantage of doing in such a way is increasing the possibility of overlap among link vectors of different web pages, thus increasing the possibility of relevant.
- Create a new web page representation by an quaternary vector of content, in-links, out-links and PageRank. Study the difference among different clustering methods of content based, link based, content-link coupled, link-PageRank coupled and content-link-PageRank combined.
- Propose two construction patterns of PageRank vector: one is a single value

of PageRank, and the other is a 11-dimensional vector with the item located by the PageRank value assigned to be 1. We try to figure out the effect difference of these two PageRank constructions on the clustering.

- Study the effect of content, in-links, out-links and PageRank on clustering. A series of weight assignments are carried out in clustering trying to find out the optimized weighting of different elements.

1.4 Outline

The rest of the report is organized as follows:

- **Chapter 2 - Background and Related Work** will introduce the basic knowledge of document clustering by describing the document representation based on vector space model and TFIDF weighting scheme, as well as K-Means clustering and the evaluation metrics of cluster quality. It will also introduce the basic web graph theory of link analysis and PageRank algorithm. Some related work of content based clustering, link based clustering and content-link combined clustering on both normal document collection and hypertext collection will be explored. In the end of this chapter, some open source framework or tools used in this project, such as Apache Lucene, Apache Mahout and etc, will be described briefly.
- **Chapter 3 - Clustering Design** will describe our authoritative K-Means clustering method for web search results, which is based on the web page content, hyper links and PageRank. First, the basic idea of how to construct term, in-link, out-link and PageRank vector will be explained in detail. Second, it gives a brief anotomy of our authoritative K-Means clustering system followed by the description of functionalities of each component. Finally, different vector combinations to represent web page will be described.
- **Chapter 4 - Experiments and Evaluation** will present the experiments details including the software/hardware configuration, experiment methods and test data collections. The results will be evaluated and discussed to reveal the difference among different clustering methods, the difference between different types of PageRank vector and the effect of different types of element vector.
- **Chapter 5 - Conclusion and Future Work** will give some concluding remarks of the experiment results and the suggestions for future work.

Chapter 2

Background and Related Work

In this chapter, we will cover some basic knowledge and technologies related to this project. First, we introduce the concept of information retrieval model, describe the vector space model in detail. Then, we introduce document clustering and mainly focus on K-Means clustering on which our authoritative clustering algorithm is based. We combine the in-link, out-link and Google PageRank into our clustering algorithm, thus some knowledge of link analysis will also be introduced in this chapter. We use several open source frameworks and tools to implement our algorithm, so it is helpful to introduce them here to help readers understand why they are selected and how they are used. Finally, we will give a short exploration of previous researches by others on combining authoritative information into document clustering, specially in web page clustering.

2.1 Information Retrieval Models

Two central problems regarding information retrieval (IR) are the representation of document and the notion of document relevance. The documents stored in the computer can exist in various formats, such as text, images, video/audio clips and etc. Different formats lead to their own logic representations of documents, and then different notions of document relevance. Text document is composed of a sequence of words/terms basically, and therefore the text document relevance could be decided by the degree of overlap on the sequences of terms representing different documents. User query can also be treated as a normal text document. The diversity of document representation and relevance notion yields distinct IR models.

The purpose of this section is to provide a conceptual basis for the document clustering study. We first propose a brief taxonomy of IR models. Second, we present the vector space model in detail, which is the most relevant IR model (also used in text categorization) referring to our project. Last, we introduce several weighting schemes, and emphasize the scheme of term frequency and inverse document frequency (TFIDF).

2.1.1 A Taxonomy of IR Models

As given the definition by Ricardo Baeza-Yates and Berthier Ribeiro-Neto [1], an IR model is a quadruple $\{$D, Q, F, R(q$_i$, d$_j$)$\}$ where

1. **D** is a set composed of logical views (or representations) for the documents in the collection.
2. **Q** is a set composed of logical views (or representations) for the user information needs. Such representations are called queries.
3. **F** is a framework for modeling document representations, queries, and their relationships.
4. **R(q$_i$, d$_j$)** is a ranking function which associates a real number with a query q$_i$ $\in$ **Q** and a document representation d$_j$ $\in$ **D**. Such ranking defines an ordering among the documents with regard to the query q$_i$.

Document representation is the fundamental part of IR model, and the framework derives the relevance calculation and ranking algorithm from the document representation. Document representations are what we can retrieve, queries are what we want to retrieve, and framework including ranking function is how we retrieve information. For instance, for vector space model, the documents and queries are represented as the n-dimensional term vectors, and the framework is composed of the vector representations and standard linear algebra operations on vectors.

The IR models are categorized within two dimensions: the mathematical basis and model properties [2]. In mathematical basis dimension, the IR models are categorized into four groups: *Set-theoretic models*, *Algebraic models* and *Probabilistic*. *Set-theoretic models* just as its name implies are based on set theory. Documents are represented as sets of terms in these kinds of models, and document relevance calculation is usually derived from set-theoretic operations, such as AND, NOT, UNION and etc. *Algebraic models* generally represent documents and queries as vectors, matrices, or tuples. The similarity between documents is based on some linear algebra operations on vectors. In *Probabilistic models*, the framework for modeling documents and queries representations is based on probability theory. Similarities of documents are computed as probabilities of that they are relevant each other. In properties dimension, the IR models are categorized into three groups : *Models without term-interdependencies*, *Models with immanent term interdependencies* and *Models with transcendent term interdependencies* [2]. Figure 2.1 shows the detailed categorization of IR models.

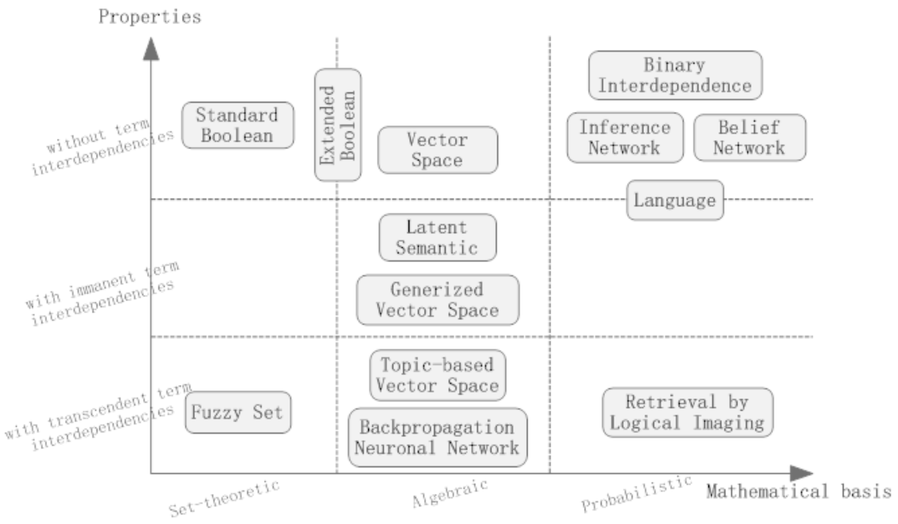


Figure 2.1: IR models categorization

The IR models in which each document is represented as a set of index terms are called classical IR models. An index term is simply a word (or a phrase in some cases) that occurs in the text document. It is generally the smallest semantic unit that constructs the document and helps to express the main theme of document. It is natural to use all distinct index terms of one document to make up its logical view, such one is called *full text*. However, there is a more flexible and effective way to form the index terms set: keep those more meaningful terms and remove the others. In general, nouns are the most meaningful compared to adjectives, adverbs, connectives and etc, e.g. “a”, “the” and “in” are almost useless to help summarize the document but just work as complements. By using such a way to represent the document, it can not only reduce the size of index terms set and thus improve the efficiency of information retrieval, but also increase the precision of information retrieval. Given a set of index terms for a document, it is obvious that not all terms contribute to describe the document content equally. Some terms could be vaguer than others, and some terms may have higher frequency of occurrence. This effect is captured through the assignment of numerical weights to each index term of a document. Therefore, some classic models assign a binary value for each index term to indicate the occurrence or not, just as what standard boolean model does, and some assign a term frequency instead as what vector space mode does. More detailed description of weights will be presented in later section.

2.1.2 Vector Space Model

Vector space model is an algebraic model for representing text documents (and any objects, in general) as vector of identifiers, such as index terms [3]. It recognizes that the mere use of binary value in standard boolean model is insufficient to perform the information retrieval, and therefore, it proposes a framework to provide partial matching by assigning non-binary values/weights for index terms in the vector. These term weights are used to compute the degree of similarity between document and user query (For document clustering, the similarity computation between documents is required). After computing all similarities between user query and documents, the retrieved documents are able to be sorted in a decreasing order of these similarities. The higher similarity to user query the document has, the more relevant it is to user query. Vector space model considers the documents that match the query terms partially, which results in that the retrieved documents list is much more precise than the one retrieved by standard boolean model. The definition of vector space model is given as follows:

Definition *Suppose D = $\{d_1, d_2, ..., d_N\}$ is the documents collection where N is the collection size, K = $\{k_1, k_2, ..., k_T\}$ is the index terms dictionary where T is the total number of index terms existed in all documents and query. Associate a positive and non-binary weight $w_{i,j}$ with a pair (k_i, d_j), and further, the index terms in the query are also weighted by associate $w_{i,q}$ with the pair (k_i, q), where q is the query and $w_{i,q} \geq 0$. Then, the query vector $\vec{q}$ = ($w_{1,q}$, $w_{2,q}$, ..., $w_{T,q}$), the document vector $\vec{dj}$ = ($w_{1,j}$, $w_{2,j}$, ..., $w_{T,j}$), where j $\in$ (1, 2, ..., N).*

The vector space model procedure is composed of three stages: document indexing, term weighting and similarity computation [4]. In document indexing stage, both full text indexing and partial text indexing can be adopted, but the later one is preferred because the full text indexing not only increases the vector space size and thus reduce the efficiency of information retrieval but also introduces a lot of noisy terms (non significant terms) that may influence the result of information retrieval.

Several techniques can be applied for partial text indexing, one is called stop list that contains all noisy terms, such as adverbs, connectives, prepositions and etc (e.g. “the”, “on”, “and”). Each document to be indexed will be filtered using the stop list, every stop word will be removed from the document and only significant terms can be kept as index terms. In general, 40-50% of the total number of words in a document can be removed with the help of stop list. The second one is called stemming, which is a linguistic morphology concept. Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form - generally a written word form [5]. For instance, the words “cats” and “cat” can be treated as their stem “cat”, the words “buy”, “buying”, “bought” and “buys” can be treated as “buy”. Even more advanced, the stemming can consider the synonyms, such as “car” and “vehicle”. There are several types of stemming algorithms such as *Brute Force Algorithm*, *Suffix Stripping Algorithm*, *Lemmatisation Algorithm* and etc. The partial text indexing can also be based on the term frequency, where terms that have both high and low frequency (a high or low frequency shreshold could be set) within a document are considered to be index terms. However, this technique is difficlut to implement in automatic indexing in practice.

After indexing, it needs to assign the weight for each index term of each vector. The index term vectors assigned with weights are the final logical representations of document or query, which are prepared for the similarity computation. There are three main factors affecting the term weighting: term frequency (TF), collection frequency and document length. These three factors can be operated to make the resulting term weight. Term frequency is somewhat content descriptive for the documents and thus can be used directly as the weight for simplification, although the result using this kind of weight might not be good. Collection frequency is a factor trying to discriminate one document from the other. Among all these kind of factors, inverse document frequency (IDF) assuming that the importance of a term is proportional with the number of document term appears in, is the most popular one. The third possible weighting factor is a document length that is often used for the normalization of term vector. Long documents usually have a larger term set than short documents, which enables long documents to be retrieved with higher probability than short documents. The weighting scheme will be described detailedly in next section.

Take such a scenario shown in Figure 2.2 as an example: There are three documents and one query, d_1 = “the cat is an animal”, d_2 = “cat likes to eat mouse”, d_3 = “the mouse is usually small, mouse is the thief” and q = “cat and mouse”. After indexing with the help of stop list and weighting by term frequency, the documents and query can be represented as the vectors as shown in Table 2.1.

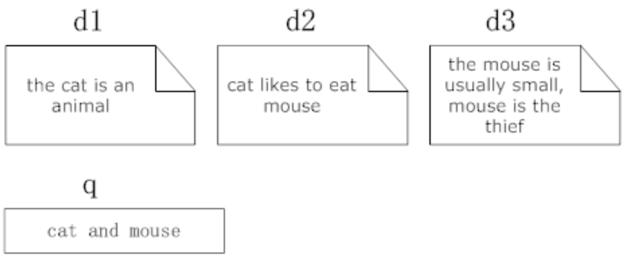


Figure 2.2: Information retrieval scenario.

The third stage is similarity computation. The similarity is determined by using associative coefficients based on the inner product of the document and query

Table 2.1: Term vectors for documents and query with term frequency

	cat	animal	like	eat	mouse	small	thief
d1	1	1	0	0	0	0	0
d2	1	0	1	1	1	0	0
d3	0	0	0	0	2	1	1
q	1	0	0	0	1	0	0

vectors, where term overlap indicates similarity. Cosine coefficient is the most popular similarity computation method, which measures the angle between the document and query vectors as shown in Figure 2.3.

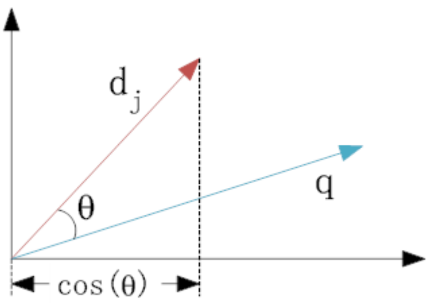


Figure 2.3: The cosine coefficient is adopted as similarity computation

The smaller the angle between two vectors is, the higher similarity these two vectors have. For instance, if the angle is 90 degree and cos(90)=0, there is no overlap between these two vectors at all, which means they are not relevant completely. If the angle is 0 degree and cos(0)=1, these two vectors are totally overlapped, which means they reach the maximum of similarity because they can be seen as equal. Equation 2.1 is the cosine similarity computation, where $|\vec{dj}|$and $|\vec{q}|$are the norms of the document and query vectors that provide a normalization in the space of the documents, T is the total number of index terms, j $\in$ {1, 2, ..., N} and N is the document collection size.

$$\begin{aligned} sim(d_j, q) &= \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|} \\ &= \frac{\sum_{i=1}^{T} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{T} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{T} w_{i,q}^2}} \end{aligned} \tag{2.1}$$

Because the weight is a non-binary positive value and similarity varies from 0 to 1, the vector space model ranks the documents by their degree of similarity to the query, not predicting whether a document is relevant or not in standard boolean model. Term weighting, document ranking and the support of partial matching, are all the advantages of vector space model over standard boolean model. However, vector space model also has some limitations, such as long documents are poorly represented, it assumes terms are independent, the terms appearance order in document is lost in vector representation, and etc.

2.1.3 Weighting Scheme

Unlike the binary term weight in standard boolean model that 1 indicates the term appears in the document and 0 indicates the term is absent, vector space model uses the positive and non-binary value as the term weight, which improves the precision of the term vector representation of document. Term frequency, collection frequency (e.g. inverse document frequency) and document length are three main factors affecting term weight.

Term frequency (TF), which is the number of occurrences of the term in document, is usually referred to as the *tf* factor that provides one measure of how well that term describes the document. It is the simplest approach to assign the TF as the weight for index term, just as Table 2.1 shows. Terms are the content descriptive units that constitute the document, so the frequency of term could indicate the main theme of document more or less. Intuitively, if one term (not stop word, usually nouns) appears in the document many times, then that document might be quite relevant to that term. Still back to the scenario shown in Figure 2.2 and look at their vector representation expressed in Table 2.1, the term “mouse” appears twice in document d_3, only once in document d_2 and none in document d_1, so the document d_3 might be more relevant to “mouse” than document d_1 or d_2.

In general, it is rare to use TF as index term weight directly. There are some variants of TF weight used in practice, such as the two shown in Equation 2.2, Equation 2.3 and Equation 2.4. $tf_{i,j}$ is TF weight of *i*th term in *j*th document and $freq_{i,j}$ is the number of occurrence of the *i*th term in *j*th document. In Equation 2.2, the denominator is the maximum of term frequency computed over all terms in *j*th document, and in Equation 2.3, the denominator is the sum of number of occurrences of all terms in *j*th document. Both TF weight variants are using normalization of term frequency to prevent a bias towards longer documents that may have a higher term frequency regardless of the actual importance of that term in the document.

$$tf_{i,j} = \frac{freq_{i,j}}{max_l freq_{l,j}} \tag{2.2}$$

$$tf_{i,j} = \frac{freq_{i,j}}{\sum_k freq_{k,j}} \tag{2.3}$$

$$tf_{i,j} = 1 + \log freq_{i,j} \tag{2.4}$$

Inverse document frequency (IDF) is usually incorporated to diminish the weight of those terms that occur very frequently in the document collection and increase the weight of terms that occur rarely. There is one critical problem while using raw term frequency as weight: all terms are assumed to be equally important when it comes to assessing relevancy on a query [6]. In fact, certain terms that appear in many documents have little or even no discriminating effect on distinguishing the relevance. For instance, if all documents in the collection contains one same term "cat", then this term has nearly no effect on distinguishing the relevance of documents each other because all documents have it. Therefore, the IDF appears to weaken the effects of terms that occur too often in document collection. Equation 2.5 is the conventional computation of IDF, where N is the total number of documents in the collection and n$_i$ is the number of documents that contain the term t$_i$. If the term does not occur in any document, then n$_i$ = 1. Equation 2.6 and Equation 2.7 are two variants of IDF.

$$idf_i = \log \frac{N}{n_i} \tag{2.5}$$

$$idf_i = \log(1 + \frac{N}{n_i}) \tag{2.6}$$

$$idf_i = \log(\frac{N}{n_i} - 1) \tag{2.7}$$

TF-IDF score is an effective weighting scheme trying to balance the effects of TF and IDF. Equation 2.8 is the most common computation of TF-IDF score. There are also many variants of TF-IDF score.

$$w_{i,j} = tf_{i,j} \times idf_i \tag{2.8}$$

Document length is usually used for normalizing the term weight or normalizing the cosine similarity calculation. The motivation of normalizing by document length is because long documents may have higher term frequencies and more terms, which increases the number of matches between documents (query can also be treated as document)thus long documents are more likely to be retrieved, and normalization weakens the impact of long documents. Equation 2.9 is the computation of document length, where T is the total number of terms in term vector. Equation 2.1 is an example of normalized cosine similarity, and Equation 2.10 is the normalized TF weight. By normalizing the TF weight in Table 2.1 by document length, the term vector representations are changed to the ones shown in Table 2.2.

Table 2.2: Term vectors with normalized term frequency weight

	cat	animal	like	eat	mouse	small	thief
d1	1/2	1/2	0	0	0	0	0
d2	1/4	0	1/4	1/4	1/4	0	0
d3	0	0	0	0	2/6	1/6	1/6
q	1/2	0	0	0	1/2	0	0

$$|d_j| = \sqrt{\sum_{i=1}^{T} w_{i,j}^2} \tag{2.9}$$

$$tf_{i,j} = \frac{freq_{i,j}}{|d_j|} \tag{2.10}$$

2.2 Document Clustering

Clustering, similar to classification, is a process that organizes the objects into some groups whose members are similar, such groups are also known as clusters. The big difference of clustering to classification is its unsupervised feature. Classification is supervised, the classes or groups are pre-defined and a certain classifier needs to be formed by a set of labeled objects (known as training objects) for future classification. Clustering is unsupervised, which means there are no pre-defined classes and it needs to discover and develop classes from objects through iterated clustering process.

In this section, we would like first to introduce a basic principle of clustering and the functionalities of its components. Second, an brief taxonomy of clustering is presented. Third, we describe a used widely clustering algorithm K-Means, which is used in our project. A short introduction of evaluation metrics for clustering is given at the end of this section.

2.2.1 Basic Principle of Clustering

The main goal of clustering is to divide a set of objects into several clusters to gather similar objects and scatter dissimilar objects. The basic principle of clustering is shown in Figure 2.4, where several primary components are listed.

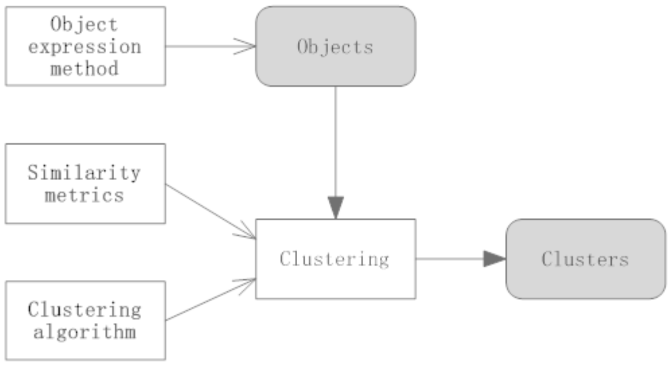


Figure 2.4: Basic principle of clustering

In general, there are three key components: objects expression method, similarity metrics and clustering algorithm. The objects themselves are usually obscured and not suitable to be clustered directly, or the direct clustering on objects is inefficient. Therefore, the objects are always expressed as other formats that are easier to be processed, e.g. the term vector representation of document in vector space model. Clustering is actually performed based on these kinds of object expression. Similarity metrics is the second key component of clustering, which provides the ability to compute the similarity between objects or between objects and clusters.

Distance measure is a quite widely used similarity metrics in document clustering. There are many different distance measures, e.g Euclidean distance measure, Manhattan distance measure as shown in Figure 2.5. Figure 2.6 shows several simple distance measures specially between cluster centroid and object. The clustering algorithm specifies the clustering process, such as how to assign objects to clusters according certain similarity metric, when the clustering should be ended. There might be some other components such as labeling the clusters to make them much more semantical and etc.

2.2.2 A Taxonomy of Clustering

There are many different clustering techniques, such as exclusive clustering, overlapping clustering, hierarchical clustering and probabilistic clustering and etc. Figure 2.7 shows some examples of different clustering techniques. Exclusive clustering just as its name implies can assign an object to only one cluster and it is not allowed that one object belongs to more than one cluster. K-Means is one of this kind of clusterings. The overlapping clustering allows the object to be clustered

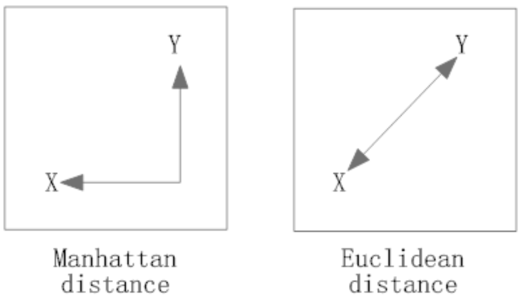


Figure 2.5: Example of Manhattan and Euclidean distance measure.

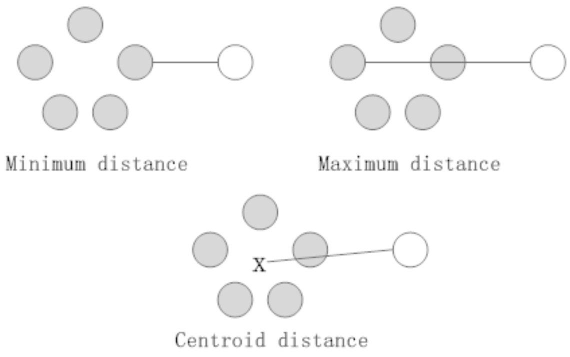


Figure 2.6: Different distance measures between object and cluster.

into several groups. Fuzzy C-Means is one representative of this kind of clustering. Hierarchical clustering is based on the union of similar clusters, the union process will be iterated until the pre-defined number of final clusters is reached. Probabilistic clustering totally uses probabilistic approaches, such as Mixture of Gaussians.

2.2.3 K-Means Clustering

K-Means clustering, first developed by J. MacQueen in 1967, is one of the most common clustering algorithms. It clusters each data point into one of K groups. K is a pre-determined positive integer that can be obtained by arbitrary selection or by some other training processes that observe the data relationships iteratively. Once the number of final clusters is decided, it needs to pick up K data points from data collection as the initial centroids for the first assignment of data points. The assignment of all data points to different clusters is performed iteratively until

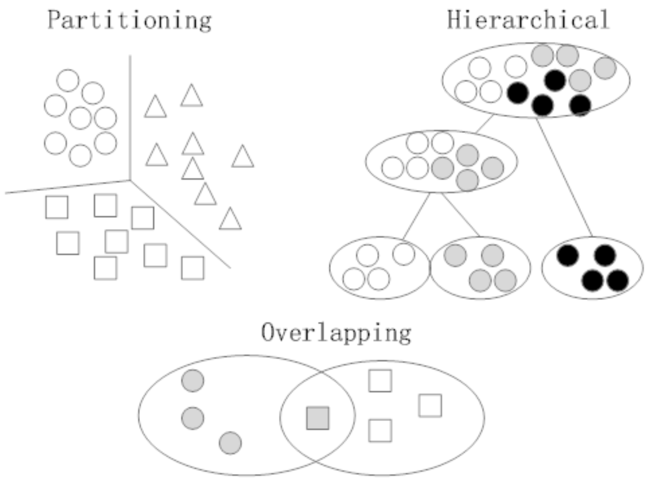


Figure 2.7: Examples of different clusterings.

some stop condition is reached. The main principle of K-Means is described as follows:

1. Pre-determine the *K* number of final clusters and randomly select the K data points as initial cluster centroids.
2. Assign each data point to the cluster that is closest to.
3. Recompute K centroids after all data points have been assigned to corresponding clusters.
4. Repeat the step 2 and 3 until the some stop condition is reached, e.g. the certain amount of iteration is finished or all cluster centroids don't change any more between iterations and etc.

Distance measure is usually the most common similarity metrics K-Means clustering uses, such as Squared Euclidean distance measure as shown in the Equation 2.11, where x$_1$, x$_2$, ..., x$_n$ is the representation of point *X* and y$_1$, y$_2$, ..., y$_n$ is the representation of point *Y*. But both Euclidean distance and Squared Euclidean distance don't consider the normalization, therefore, K-Means clustering uses cosine similarity metrics that is described previously in the section of “Vector Space Model”.

$$d = \sum_{i=1}^{n}(x_i - y_i)^2 \tag{2.11}$$

2.2.4 Evaluation Metrics

For clustering, some measures (also known as evaluation metrics) of clusters quality are required to evaluate the clustering methods. There are basically two types of evaluation metrics: internal quality measure and external quality measure. The internal quality measure compares different sets of clusters without reference to external knowledge, e.g. entropy, purity, F-measure and etc. The external quality measure evaluates how well the clustering is by comparing the clusters produced by the clustering methods to known classes, e.g. high intra-cluster similarity and low inter-cluster similarity.

2.3 Link Analysis

As one primary element of web page, hyperlinks contain a great number of information about web page itself, both structural and semantical (e.g. anchor text). Hyperlinks always lead to some web pages (even leading to themselves) and those hyperlinks connecting to other web pages are usually viewed as a conferral of authority. Thus one simple link analysis can be performed by measuring the number of in-links of a web page to evaluate its quality. However, because there are some kinds of hyperlinks just connecting to themselves or sponsor links, which can not promote the web page quality, thus more sophisticated link analysis must be evolved. The analysis of hyperlinks has been conducted many years ago and applied into the web search. Most commercial web search engines like Google, Yahoo and etc, involve the hyperlinks analysis trying to improve their ranking of web search results. In such kind of ranking algorithms, hyperlinks are utilized to compute a composite score for a web page on different queries. In addition, link analysis is also used in web crawling and clustering.

The purpose of this section is trying to introduce some basic knowledge of link analysis, such as web graph theory, to help readers understand what is in-links and out-links, how they are used in link analysis. We also describe the PageRank algorithm that is relevant to our project.

2.3.1 Web Graph

We all know that the most primary feature of the real web is its ability of dynamic growing, hundreds and thousands of new web pages or other information segments are joined into the web every moment. However, if we view the web statically, it consists of certain amount of static web pages and hyperlinks that connect the all static web pages each other. In such a way, the web is like a directed graph in which the web pages are nodes and hyperlinks are directed edges. Figure 2.8 depicts a sample of small web graph. In this small web graph, there are six nodes

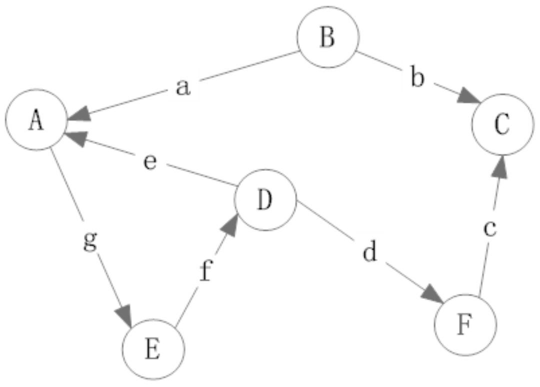


Figure 2.8: A sample of small web graph

labeled from *A* to *F* that representing six web pages and seven directed edges labeled from *a* to *g* representing seven hyperlinks connecting the web pages. This web graph is not strongly connected because not all pages can be reached by others, e.g. page *B*. The hyperlinks pointing from one page to other pages are called out-links of that page, e.g. page *B* has two out-links with *a* pointing to page *A* and *b* pointing to page *C*. The hyperlinks by which one page is pointed are called in-links of that page, e.g. page *A* has two in-links with one from page *B* and the other from page *D*.

The number of out-links and in-links of one page are called "outdegree" and "indegree" of that page. "outdegree" and "indegree" are two of important parameters to determine the popularity of web page. For instance, those web pages that have many in-links can be accessed more frequently. A normal hyperlink usually consists of two parts: an URL address and the anchor text. URL stands for Uniform Resource Locater, which is a global address of documents or any other resources on World Wide Web. It is generally encapsulated in the "href" attribute of the "<a>" tag (HTML code). The anchor text is the text surrounded by the "<a>" tags. Anchor text plays quite big role in link analysis for web search and clustering. Since it looks just like a normal text, some people assign anchor text to the content of web page where the anchor text is from. However, some others assign it to the web page where its binded URL points to because they think the anchor text usually provides a small summary of target page.

Some researchers propose a bowtie concept of directed web graph [7] as shown in Figure 2.9. They provides three main categories for web pages: IN, OUT and SCC. As described in [1], a web surfer can pass from any page in SCC to any page in OUT, but can not pass from any page in OUT to any page in SCC; the surfer can pass from any page in IN to any page in SCC and consequently to any page in OUT, but can not pass from any page in SCC to any page in IN; the surfer

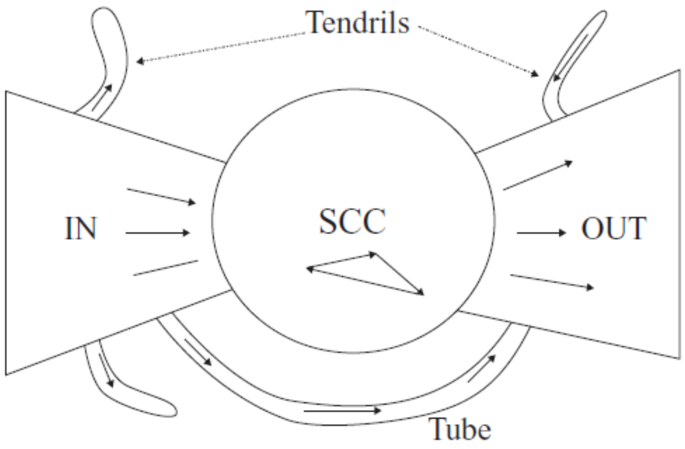


Figure 2.9: The bowite structure of the web.

can pass from any page in SCC to any page in SCC. IN and OUT have roughly equal size, but SCC is larger. Most web pages belong to one of these three categories, and there are still small portion of web pages formed tubes outside SCC that lead directly from IN to OUT. The last kind of pages are called tendrils that either lead nowhere from IN or from nowhere to OUT.

One more concept we would like to introduce is the neighborhood. Neighbors of one page are all those pages connected with that page by either in-link or out-link. The neighborhood concept is utilized by some researchers in document clustering [8]. Figure 2.10 shows an example of effect of neighborhood in clustering. Although page *F* is first assigned to cluster *B*, it has only one neighbor in cluster *B* but three neighbors in clustering *A*, therefore, the page *F* has high possibility belonging to cluster *A*.

2.3.2 PageRank

The number of in-links of one page can vary quite great among different web pages. As mentioned previously, the web pages having more in-links will have higher possibility to be accessed, thus more “important ” than those pages having few in-links. The count of in-links indeed indicates the importance of web page somewhat, however, it is not always correct because some web pages that pointed by other important pages also might be important even they have few in-links. For example, the web pages pointed by Yahoo home page are usually important. Therefore, weighting equally over all in-links is not accuracy, which means different in-link will have different weights, e.g. the links of some famous

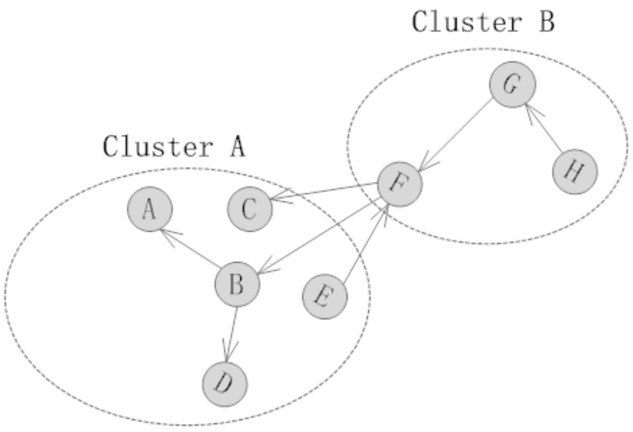


Figure 2.10: Effect of neighborhood in clustering.

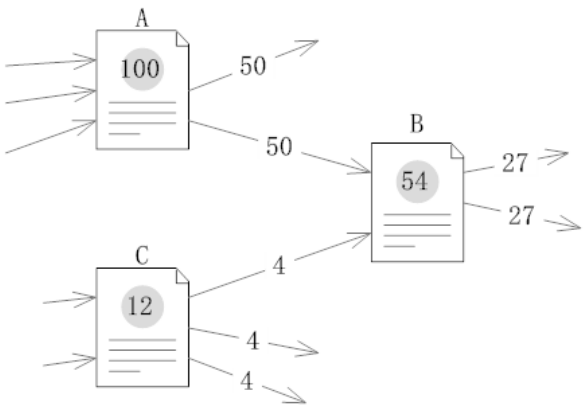


Figure 2.11: Simplified propagation of ranking through links.

and important web pages like Yahoo, Google and etc, will have higher ranks than the other pages. PageRank [9][10] is such an attempt trying to combine both the number and the rank of in-links for the approximation of “importance”.

The basic idea of PageRank is the propagation of ranking through links. The rank of a web page will be allocated to its out-links equally, and a web page will accumulate all ranks from its in-links. Figure 2.11 shows such an propagation, page *A* allocates its rank of 100 to two out-links equally, and page *B* accumulates the ranks of its two in-links and allocates to its out-links equally. The PageRank of p_j can be computed by the Equation 2.17:

$$PR(p_j) = \frac{1-d}{N} + d\sum_{i=1}^{n} \frac{PR(q_i)}{L(q_i)}, j \in [1, N] \tag{2.12}$$

Where *d* is a damping factor usually set to 0.85, *N* is the number of all web pages that compute the PageRank, q_i is the in-link of page p_j, *n* is the indegree of page p_j and *L(*q_i*)* is the outdegree of page q_i.

Although the computation of PageRank by Equation 2.12 is possible, it is usually computed through the computation of the principal left eigenvector of transition probability matrix in certain amount of iterations, which is defined in Equation 2.13 combined with Equation 2.14 and 2.15.

$$R^{(n+1)} = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ ... \\ (1-d)/N \end{bmatrix} + d \begin{bmatrix} \ell(p_1,p_1) & \ell(p_1,p_2) & ... & \ell(p_1,p_N) \\ \ell(p_2,p_1) & ... & ... & ... \\ ... & ... & \ell(p_i,p_j) & ... \\ \ell(p_N,p_1) & ... & ... & \ell(p_N,p_N) \end{bmatrix} R^{(n)} \tag{2.13}$$

$$R = \begin{bmatrix} PR(p_1) \\ PR(p_2) \\ ... \\ PR(p_N) \end{bmatrix} \tag{2.14}$$

$$\sum_{i=1}^{N} \ell(p_i, p_j) = 1 \tag{2.15}$$

Where the adjacency function ℓ(p$_i$,p$_j$) is 0 if page p_j doesn't link to p_i.

2.4 Related Work

Web search results are the target collection that we would like to perform our clustering. Unlike the normal text documents that only contain the plain text, or the academic papers that besides the plain contents, they also contain some authoritative information such as conference, citation, reference and etc, the web pages have a unique feature over those two kinds of documents mentioned above: the hyperlinks, both in-links and out-links. The Google PageRank is also derived from the hyperlink structure in essential. Therefore, the clustering of web pages only based on the content is obviously insufficient, and it could be predicted that the hyperlinks can undoubtedly improve the web clustering. Many researchers did a lot of work to study the characteristics and structure of hyperlinks of web pages, design the clustering algorithm based on hyperlinks and evaluate their effects on web clustering.

2.4.1 Link-based IR

Sergey Brin and Lawrence Page [9][10] develop a new algorithm called PageRank, and Jon M. Kleinberg [11] develops another algorithm called Hyperlink-Induced Topic Search (HITS), also known as Hubs and authorities. Both of these two algorithms are based on hyperlinks analysis to help rating the web pages. The detailed description of these two algorithms are covered in *Link Analysis* section. Monika Henzinger [12] gives a short survey of the use of hyperlink analysis for ranking search results in web search engines. She classifies the hyperlink-based ranking algorithms into two categories: query-independent algorithms that assign a score to each page independent of a specific query, and query-dependent algorithms that assign a score to each page dependent of a specific query. She mainly introduces the PageRank as one of the query independent clustering algorithms and HITS as one of the query dependent clustering algorithms.

Taher H. Haveliwala [13] proposes a topic-sensitive PageRank to be used in web search results ranking. In general, the generic PageRank is combined with IR scores to rank the web search results, but the generic PageRank doesn't capture more accurately with specified query topic since it is usually query-independent. The topic-sensitive PageRank is a kind of personalized PageRank in essential. The basic formula to compute the generic PageRank is given in Equation 2.16:

$$\begin{aligned} \vec{Rank} &= M' \times \vec{Rank} \\ &= (1-a)M \times \vec{Rank} + \alpha\vec{p} \end{aligned} \tag{2.16}$$

Where $\vec{Rank}$ is a rank score vector whose *i*th item indicates the *i*th web PageRank score, M is the stochastic matrix corresponding to the directed graph G of the web (if there is a link from page *j* to page *i*, then $M_{i,j}$ = $1/N_j$, N_j is the number of out-links of page *j*. The other matrix entry is 0),α is a damping factor and $\vec{p}$ is some source of rank that is the key to creating topic-sensitive PageRank. The $\vec{p}$ for generic PageRank equals to $[1/N]_{N\times1}$, which means $\vec{p}$ is a N-dimensional vector with all the items equal to 1/N. N is the size of data collection.

To create topic-sensitive PageRank for a web document *d*, it can assign different value to $\vec{p}$. In addition, the selection of damping factor α is another way to create topic-sensitive PageRank. ODP-biasing (16 representative basis topics taken from the Open Directory) is used to replace $\vec{p}$ with ν_j, the first step is create a biased PageRank source by the Equation 2.17, where T_j is the set of pages in the ODP category C_j and $|T_j|$ is its size.

$$\nu_{j,i} = \begin{cases} \frac{1}{|T_j|} & \text{if } i \in T_j \\ 0 & \text{if } i \notin T_j \end{cases} \tag{2.17}$$

Then, the topic-sensitive PageRank of web document *d* is not one vector but a composition of 16 vectors. The second step is to calculate the query-sensitive importance score S_{qd} by Equation 2.18:

$$S_{qd} = \sum_j P(c_j|q^{'}) \cdot rank_{jd} \tag{2.18}$$

Where $P(c_j|q^{'}) = P(c_j) \cdot \prod_i P(q_i^{'}|c_j)$ and $P(q_i^{'}|c_j)$ is easily computed from the class term-vector D_j, $P(c_j)$ is chosen to be uniformed (1/16). $q^{'}$ is the context of query *q*, and equals to *q* when ordinary query is performed (There is another query scenario: search in context. The user is viewing a document and selects a term from the document for which he would like to search more relevant information).

Taher H. Haveliwala's research [13], he develops two similarity measures used for ranking comparison that are denoted as *OSim* (Equation 2.19) and *KSim* (Equation 2.20)respectively.

$$OSim(\tau_1, \tau_2) = \frac{|A \bigcap B|}{n} \tag{2.19}$$

Where τ_1, τ_2 are two rankings, *A* and *B* are two sets (the size of each set is n) corresponding to τ_1 and τ_2 respectively. *OSim* indicates the degree of overlap between the top n pages of two rankings.

$$KSim(\tau_1, \tau_2) = \frac{|(u,v) : \tau_1^{'}, \tau_2^{'}\ agree\ on\ order\ of\ (u,v), u \neq v|}{|U||U-1|} \tag{2.20}$$

Where *U* is the union of pages in τ_1 and τ_2; *u*, *v* are web pages; $\tau_1{}^{'}$ is the extension of τ_1 and it contains (*U* - τ_1) appearing after all pages in τ_1; $\tau_2{}^{'}$ is in the similar fashion.

In Taher H. HHe uses the latest web crawl from the Standford WebBase containing roughly 120 million pages as the data source. His crawl contains roughly 280,000 pages in the ODP and he uses 35 queries to perform two rankings based on generic PageRank and topic-sensitive PageRank respectively. He also discusses the effect of damping factor α and made α = 0.25. In single query scenario, he runs two rankings and conducted a user study, in which some volunteers were chosen to ask which ranking list has more relevant pages. The precision is also chosen as a evaluation metric. Finally, Taher H. Haveliwala discovers that the topic-sensitive PageRank improves the ranking significantly.

2.4.2 Link-based Clustering

Since the hyperlink is one of the main components of web document, some other researchers studied the link-based clustering intuitively. Xiaodan Zhang [14] et al. use two clustering algorithms: Content-based clustering (e.g. spherical K-Means and model-based K-Means) and Link-based clustering, which is actually the combination of previous content-based clustering procedure and iterative relaxation labeling algorithm [8], to examine the impacts of different linkage types. They study on three types of linkages:

- Explicit link, e.g. hyperlinks and citation links.
- Implicit link, e.g. co-authorship links.
- Pseudo link, e.g. content similarity links.

The description of 5-step link-based clustering algorithm is given as follows:

1. Repeat content-based clustering such as spherical K-Means or model-based K-Means clustering until it reaches a fix point.
2. Initialize document assignment *C* using output class label assignment from step 1.
3. Model re-estimation: λ_i = arg max$_{\lambda_i} \sum_{\tau c_i} \log(\tau_i|\lambda)$.
4. Iteration labeling: c$_i$ = arg max$_{c_i}$ $\log\{$Pr(N$_i$|c$_i$)Pr(τ_i|c$_i$)Pr(c$_i$)$\}$.
5. Stop until a pre-defined iteration number is reached or if *C* doesn't change, otherwise go to step 3.

C is the document class assignment, e.g. *C* = $\{$c$_1$, c$_2$, ..., c$_n\}$ and c$_i \in \{$1, 2, ..., k$\}$, *k* is the desired number of clusters; λ is cluster models, e.g. λ = $\{\lambda_1$, λ_2, ..., $\lambda_n\}$; N$_i$ represents all known class labels of the neighbor documents of document *i*; T = $\{\tau_1$, τ_2, ..., $\tau_n\}$ representing the entire collection of text of corresponding data set, *n* is the number of documents in the data set. Pr(N$_i$|c$_i$) = $\prod_{d_j \in I_i}$Pr(c$_j$|c$_i$, e$_{j \to k}$)$\prod_{d_k \in O_i}$Pr(c$_k$|c$_i$, e$_{i \to k}$), I$_i$ and O$_i$ are in-neighbors and out-neighbors of document *i*.

Eight data sets are used for the experiments: WebKB4, CORA7, CORA18, DBLP3, TDT2_10(from TREC), LATimes10, Reuters10 and 20NG. F-score, purity and NMI are three evaluation metrics used to evaluate the final clusters quality generated by content-based clustering and link-based clustering. They finally discover that using hyperlinks, citation links and co-authorship links, link-based clustering exhibits significant improvement over content-based clustering, and co-authorship links based clustering achieves the best performance, citation links based stands in the middle and the worst one is hyperlinks based clustering (pseudo links have no positive improvement). In addition to the comparison of three different types of links, Xiaodan Zhang et al. also find:

- Uniform priors is better than empirical priors for clustering (use uniform priors to approximate Pr(c_i)).
- Out-neighbors have more impacts on clustering than in-neighbors (this was evaluated by the comparison of different types of neighborhoods, e.g. in-neighbors only, out-neighbors only, combination of in-neighbors and out-neighbors, in-neighbors and their in-neighbors only and etc. Here in-neighbors of document *i* are the documents pointed to document *i* and out-neighbors of document *i* are the documents pointed by document *i*).
- Thresholding and scaling have no positive effects on clustering (thresholding is used to filter our-links between two documents whose similarity value is below this pre-defined thresholding).

Yitong Wang and Masaru Kitsuregawa [15][16] study the link-based clustering for web documents. The intuition behind their method is that pages that co-cite (share common out-links) or are coupled (share common in-links) are with high probability to be clustered into the same cluster. They extend the standard K-Means clustering algorithm that uses cosine similarity between pages or between page and the cluster centroid. Each page *P* in web search results *R* is represented as two vectors: P_{out} (N-dimensional) and P_{in} (M-dimensional), *N* and *M* are total number of all distinct out-links and in-links for all pages in *R* respectively. The similarity between page *P* and *Q* (similarity between page and cluster centroid is calculated in the same way) could be calculated by the equation 2.21.

$$\begin{aligned} Sim(P,Q) &= \frac{(P \bullet Q)}{|P||Q|} \\ &= \frac{((P_{out} \bullet Q_{out}) + (P_{in} \bullet Q_{in}))}{|P||Q|} \end{aligned} \tag{2.21}$$

$|P|=\sqrt{\sum_{1...N} P^2_{out\ i} + \sum_{1...M} P^2_{in\ j}}$, $|Q|=\sqrt{\sum_{1...N} Q^2_{out\ i} + \sum_{1...M} Q^2_{in\ j}}$. The cluster centroid *C* can be represented by the combination of C_{out} and C_{in}; $C_{out} = \frac{1}{|S|} \sum_{P_i \in S} P_{out\ i}$, $C_{in} = \frac{1}{|S|} \sum_{P_i \in S} P_{in\ i}$; $|S|$ is the size of cluster *S*.

Their 4-step link-based clustering algorithm is described as:

1. It filters irrelevant pages whose summary of in-links and out-links is less than 2 from web search results.
2. Each relevant page is assigned to existing clusters if **(a)** similarity between the page and cluster centroid is above a pre-defined similarity threshold and

(b) the page has a link in common with 30% near common links of the correspondent cluster (near common link of cluster means links shared by majority members of one cluster). If none of current existing clusters meet the demands, the page will become a new cluster itself.

3. Update cluster centroid if it changes. While one page belongs to more than one cluster, it is limited to top 10 clusters based on similarity values. The assignment is executed iteratively until the centroids of all clusters are no longer changed.
4. Final clusters set are generated by merging any two base clusters if they share majority members based on the pre-defined merging threshold.

There are three parameters in the algorithm that affect the final clusters quality: number of in-links (in order to limit the size of link structure of each page, the maximum of in-links for each page is pre-defined.), similarity threshold and merging threshold. They use four different data sets: top 750 pages from Google by the query of “Jaguar (1)”, top 750 pages from Google by the query of “Jaguar (2)”, top 200 pages from AltaVisa by the query of “Data mining” and top 400 pages from Yahoo by the query of “Java”, to evaluate the effect of similarity and merging threshold. Finally, they discover that more pages are clustered and maximum cluster size increases when similarity threshold decreases, and it can generate reasonable clusters when merging threshold is set as 0.75 or 0.8.

2.4.3 Content-link Combined Clustering

Many researchers study the document clustering based on the combination of document content and links [17][18], and some focus on the content-link combined clustering for web documents, such as Yitong Wang and Masaru Kitsuregawa [19], Dharmendra S.Modha and W.Scott Spangler [20], He XiaoFeng et al. [21]. The basic ideas of [19][20] are quite similar. The web page *P* is represented as triple vectors as below, where N, M and L is the total number of distinct out-links, in-links and terms for all web pages in the set of web search results respectively.

- P_{out} (N-dimensional) is the out-links vector of *P*. The *i*th item of vector P_{out} indicates whether *P* has the correspondent out-link as the *i*th one in N out-links. If yes, the *i*th item is 1, else 0.
- P_{in} (M-dimensional) is the in-links vector of *P*. P_{in} is defined similar to P_{out}.
- P_{kword} (L-dimensional) is the content vector of *P*. The *k*th item of vector P_{kword} indicates the frequency of the correspondent *k*th term of L appeared in the collection. P_{kword} combines all terms that exist in anchor text, snippet, meta-content and anchor window (include 2 words to the left and 2 words to the right of the anchor text).

The triple vectors are combined giving different weights and the cosine similarity between two pages or between web page and cluster centroid can be calculated by the equations below:

$$Sim(A,B) = \omega_1 \cdot OutSim(A,B) + \omega_2 \cdot InSim(A,B) + \omega_3 \cdot ConSim(A,B) \quad (2.22)$$

$$OutSim(A,B) = \frac{A_{out} \bullet B_{out}}{|A_{out}| * |B_{out}|} \quad (2.23)$$

$$InSim(A,B) = \frac{A_{in} \bullet B_{in}}{|A_{in}| * |B_{in}|} \quad (2.24)$$

$$ConSim(A,B) = \frac{A_{kword} \bullet B_{kword}}{|A_{kword}| * |B_{kword}|} \quad (2.25)$$

$$\omega_1 + \omega_2 + \omega_3 = 1 \quad (2.26)$$

Yitong Wang and Masaru Kitsuregawa use an extension of standard K-Means clustering and use a similarity threshold to replace the pre-defined *K* centroids to control the clustering process. The noisy pages that below the threshold will be filtered frist. After K-Means clustering, a hierarchical clustering [22][23] is used to merge some similar cluster pairs, based on previous clustering result. For instance, Sim(C$_a$, C$_b$) is bigger than HR-mering threshold (another pre-defined similarity threshold), then C$_a$ and C$_b$ will be added into C$_A$. Other cluster pairs that share any one member with C$_A$, let's say (C$_a$, C$_c$) or (C$_b$, C$_c$) will be added into C$_A$, which results in that C$_a$, C$_b$ and C$_c$ are contained in C$_A$. The process will be repeated until no cluster pairs have the similarity bigger than HR-merging threshold.

They select top 200 web search results obtained from Google search engine with 8 different queries respectively: "food", "chair", "black bear attack", "moon river", "jaguar", "big apple", "salsa" and "jordan" as the test collections to do conduct the experiments to study on:

1. Comparisons among content-based, link-based and content-link coupled clustering methods.
2. The effect of similarity threshold.
3. The effects of out-links, in-links and contents.
4. The effect of overlap (one page could belong to more than one cluster).

5. Comparison between standard K-means and the proposed clustering algorithm.

Average entropy, Average distribution, Precision and Recall are the evaluation metrics they used in their research. Finally, they discover that in-links usually affect the number of clusters produced as well as the size of the maximum cluster, out-links usually affect the percentage of pages clustered and contents affect the recall and precision of the clustering results. When giving appropriate weights for out-links, in-links and contents (e.q. the weight of out-links is less than 0.5; the weight of in-inks is less than 0.5; the weight of contents is less than 0.75 and greater than 0.25), the content-link coupled clustering reaches much better results.

Later, Yitong Wang and Masaru Kitsuregawa propose two enhancement techniques for the content-link based clustering [24]. The first technique is called *In-links reinforcement*, the purpose of which is to reduce the in-links feature vector dimension and also find the relationship between in-links. Its basic principle is described as follows:

1. Executing content-based clustering to generate *x* groups.
2. Represent each in-link page *p* as an *x*-dimensional vector, whose *k*th item indicates whether the in-link page *p* has an out-link page in *k*th cluster of *x* groups.
3. Cluster in-link pages into *y* groups based on vector similarity.
4. Map each in-link page to the cluster it belongs to and then for data set, the in-link space is reduced to *y*-dimensional.
5. Execute the content-link coupled clustering with renewed in-link vectors.

The second technique is called *anchor window analysis*, which is to differentiate the effects of content term and anchor window term by dividing term vector into two parts: anchor window term vector and content term vector. Then the content vector similarity in Equation 2.25 will be changed to Equation 2.27, where *Akword* indicates the terms appearing in anchor window and *Ckword* indicates the terms appearing in content.

$$ConSim(A,B) = \frac{1}{2} \cdot \frac{A_{Akword} \bullet B_{Akword}}{|A_{Akword}| \cdot |B_{Akword}|} + \frac{1}{2} \cdot \frac{A_{Ckword} \bullet B_{Ckword}}{|A_{Ckword}| \cdot |B_{Ckword}|} \quad (2.27)$$

2.4.4 PageRank-based Clustering

Konstantin Avrachenkov et al. propose a new PageRank-based clustering algorithm for hypertext documents (PRC) [25]. It is composed of two parts: determination of the core nodes or centroids of clusters and the assignment of nodes to clusters. There are two steps to complete the cluster centroids determination:

1. Determine a list of candidate nodes for centroids by sorting nodes in decreasing order of PageRank and Reverse PageRank product.
2. Choose the centroids from the candidate nodes. If two candidate nodes belong to the same cluster, the one with worse ranking is discarded. Decide if two candidate nodes belong to the same clustering by using a threshold on the number of one and two-step directed paths or JS divergence [26].

Once a list of centroids is formed, the node assignment is proceeded by using Equation 2.28:

$$B_s = c_s P + (1 - c_s) K_s \tag{2.28}$$

Where B_s is the possibility of node to be assigned to certain cluster; c_s is the possibility to go out of cluster *s* in the random surfer mode (in practice, it adopts 0.5); K_s is the topic-sensitive PageRank vector; *S* is the centroid index.

2.5 Open Source Framework or Tools

We use several open source frameworks or tools to implement our authoritative K-Means clustering algorithm, such as using Apache Lucene for document indexing, Apache mahout for term vector creation and K-Means clustering. Because our clustering is based on web search results and involves in-links and out-links of web page, we use HtmlParser and HtmlCleaner to help for parsing HTML web pages. Some implementation details in Google search results retrieval and Goolge PageRank calculation will be described in next chapter. Here we only introduce these open source frameworks briefly to help readers to grasp a whole picture of how they will be used in our project.

2.5.1 Apache Lucene

Apache Lucene [27] is an open source software project held by Apache Software Foundation for developing full text search engine. It is a high-performance, full-featured and scalable Java libraries set that helps developers to build their own custom search applications. In other words, it is not a complete application but specifically an text search engine API set, which means that although Lucene has done all the hard parts, some easy programming tasks to invoke those APIs are still left to the developers. Apache Lucene was originally written by Doug Cutting and came out available with the first version of 0.0.1 at SourceForge web site at the beginning. It was accepted into the Apache Software Foundation's Jakarta family of open source server-side Java products in September of 2001. From then on, more and more developers and users joined the improvement of Lucene. After the evolution for almost ten years, the latest version of Lucene so far is 3.0.1 that can be downloaded from the official website of Apache Software Foundation: http://www.apache.org/dyn/closer.cgi/lucene/java/.

Apache Lucene comprises two main features that are text indexing and searching, and also some other auxiliary features such as queries parsing and etc. The text indexing is the main part we use in our clustering of web search results. Although it is possible to sequentially scan each web document from the web search results to extract terms for term vector creation, the approach has an obvious flaw that it doesn't scale to larger search results set or cases where web pages are very large. This is the reason why we index the web document first and convert it into a certain format: to eliminate the slow sequential scanning process. The conversion process is called *indexing* and its output is called *index*.

In general, there are four components of indexing: acquire content, build document, analyze document and index document [28]. Acquire content process is usually referred to as spider or crawler, whose goal is to collect the documents and scope the contents that need to be indexed. Lucene doesn't provide any functionalities to support acquiring content, but there are many open source crawlers such as Nutch, Heritrix and etc. The second process is document building, which translates the raw content into certain format of information (it is usually called "document" that is different from conventional document) used by the search engine. Such kind of documents might consist of several fields with values, such as *title*, *body*, *url* and etc. Lucene provides some APIs to build fields and documents. The textual fields in a document can not be indexed directly, e.g. the value of *title* field of one document is "The cat is an animal" and it makes no sense to index all terms. This is the reason why the third process named analyze document is needed. The responsibility of analyze document process is to tokenize the text into a series of individual atomic terms and filter them using some special rules, such as stop list, stemming and etc. Lucene provides several built-in analyzers including those two mentioned above. Finally, the documents that are built from raw content and analyzed by certain built-in analyzer, will be added to the *index* for complete the index process.

2.5.2 Apache Mahout

Apache Mahout is an open source project held by Apache Software Foundation with the primary goal to implement a scalable machine learning java library. With scalable it means that the algorithms included in the library can be applied on reasonably large data sets. The core algorithms of Mahout library for clustering, classification and batch based collaborative filtering are implemented on top of Apache Hadoop (another open source project by Apache Software Foundation aiming to develop software for reliable, scalable, distributed computing [29]) using the map/reduce paradigm [30]. Although the implementations of algorithms based on Apache Hadoop concentrate on rapidly processing vast amounts of data in parallel on large clusters of compute nodes, the process of data on single node is still supported.

The Mahout project was started in 2008 by several people involved in the Apache Lucene community with an active interest in machine learning and a desire for robust, well-documented, scalable implementations of common machine learning algorithms for clustering and categorization [31]. The latest version of Mahout is 0.3 released in 2010 and can be downloaded from `http://www.apache.org/dyn/closer.cgi/lucene/mahout/`. Currently Mahout supports mainly four use cases [30]:

- **Recommendation mining** takes users' behavior and from that tries to find items users might be interested in.
- **Clustering** attempts to group a large amount of documents into clusters that share some similarity in topics or other information of documents. It discovers the hierarchy of large data set and reveals its inner patterns for easier understanding. Mahout provides some clustering algorithms, such as Fuzzy clustering, Canopy clustering, K-means clustering and etc.
- **Classification** decides if the unlabeled documents belong to some certain category or not. It learns from many existing categorized documents to deduce classification rules, then it is able to assign unlabeled documents to the possible correct category.
- **Frequent itemset mining** takes a set of item groups (terms in a query session, shopping cart content) and identifies, which individual items usually appear together.

2.5.3 HtmlCleaner

HtmlCleaner [32] is an open source HTML parser written in Java and the latest version of HtmlCleaner is 2.1 that was released at SourceForge web site in September of 2008. We all know that HTML is a markup language for web pages. In general, the web pages written in HTML are well structured by different HTML tags, but there are still a lot of ill-formed HTML pages existing over the Internet. Some HTML pages may have the unclosed tag pair or disorder the tag pairs. The ill-formed HTML pages are usually unsuitable for further processing. For any serious consumption of such pages, it is necessary to first clean up the mess and bring the order to tags, attributes and ordinary text. HtmlCleaner is such a parser that takes a raw HTML page as input, reorders individual elements and produces internal well-formed XML correspondingly, which becomes ready for the further processing, such as extracting attributes or text of any specified node. By default, HtmlCleaner follows similar rules that the most of web browsers use in order to create Document Object Model, but it still supports the custom tag and rule set for tag filtering and balancing by users. In short, HtmlCleaner parses ordinary HTML pages by preparing them for XML processing with XPath, XQuery and XSLT.

The conversion of HTML page into XML is completed by invoking *clean* method of HtmlCleaner. Class *TagNode* is the heart class of HtmlCleaner. In internal parsing, all elements of the HTML page including the document itself are treated as TagNode. The *clean* method is actually return a TagNode that expresses the whole document. One powerful feature of TagNode is that it supports the navigation and parsing inside the TagNode by XPath (a short XPath tutorial can be found in `http://www.w3schools.com/xpath/`), the user can locate any inner TagNode by invoking the *evaluateXPath* method of the outer TagNode. In addtion, TagNode also provides some useful methods for parsing, such as *findElementByName*, *getAttributeByName*, *getText* and etc.

2.5.4 HtmlParser

HtmlParser is a Java library used to parse HTML in either a linear or nested fashion [33]. It provides some similar funcationalities as HtmlCleaner. The latest version of HtmlParser is 2.0 available to be downloaded from SourceForge web site. The only part of HtmlParser we used in our implementation is its *StringBean* class, which provides a functionality to extract the plain content of a HTML page removing all HTML tags.

2.5.5 Yahoo! Site Explorer Web Service

As a successful commercial web search engine, Yahoo! has its own search index database that contains completed detailed information about the web pages and their graphical structures, most of which are referred as in-links and out-links. Yahoo! Site Explorer Web Service is a tool that allows users to access these information of web pages. There are four concrete services included in Yahoo! site explorer web services [34] as shown below, and we only use the **Inlink Data** service for in-links extraction of web search results in our project.

- **Inlink Data** shows the user to retrieve the information of those pages that link to a particular page or domain.
- **Page Data** shows a list of all pages belonging to a domain in the Yahoo! index.

- **Ping** and **Update Notification** allow users to notify Yahoo! of changes to their sites.

To use the **Inlink Data** service of Yahoo! site explorer, an Application ID is required. The Application ID is a User-Agent like string uniquely identifying user's application, which can be obtained after the registration at Yahoo web site. The service is limited to 5,000 queries per IP address per day. The users can set several parameters of the service, such as the url string of target page, the number of in-links to return, the starting result position to return and etc, to post their customized queries. The returned result of in-links is presented in XML format by default. The maximum of in-links can be fetched per query is 1,000.

Chapter 3

Clustering Design

In this chapter, we will describe our authoritative K-Means clustering method in detail. First of all, the definition of web page representation as multiple vectors will be given to make readers clear of how term vector, in-link vector, out-link vector and PageRank vector are contructed in our clustering. Second, we present our authoritative K-Means clustering system and explain the functionalities of its components. Finally, we design several strategies of vectors combination to represent the web page for clustering, thus to study the effect of different types of vectors.

3.1 Web Page Representation

[19] and [20] introduce the similar concept of triple-vector representation of hyperlink document: term vector, in-link vector and out-link vector. Our web page representation is still based on their concepts primarily, just change a little bit in the construction of each vector. In addition, we involve a new vector of Google PageRank. Based on these four types of vectors described above, it is possible to represent web page in any combination of these vectors, e.g. the combination of term vector and in-link vector, the combination of term vector, in-link vector and out-link vector, the combination of term vector, in-link vector, out-link vector and PageRank vector and etc. More detail about the vector combination will be covered in later section.

3.1.1 Term Vector

As the most important element of any document no matter web page, academic paper or any other types, the content always carries most theme information of the document. Therefore, the term vector plays the biggest role in document representation in vector space model. Specifically, the content terms of web page can come from its title, description and plain content. We all know that the terms from title usually have much more concentrated meaning to summarize the document topic. Therefore, it could be helpful to assign different weights for the terms coming from different parts. However, we combine the terms from title, description and plain content equally for the simplicity in our term vector construction. All the terms will be filtered by stop list to remove those meaningless terms, such as connectives, adverbs, pronouns and etc.

The dimension of term vector space (also known as dictionary) of the web page collection could still be huge even after filtering by stop list, therefore, we introduce another two factors called *Minimum Document Frequency* and *Maximum Document Percentage* to help shrink the term vector space in the vector creation procedure. *Minimum Document Frequency* indicates the minimum number of documents where certain term should exist. For instance, if the *Minimum Document Frequency* is set to two, then only the terms occur in two or more than two documents will be kept. In general, if the terms occur in quite few documents, then the matrix of term vectors will be quite sparse, which could reduce the efficiency of computation of documents relevancy because the vector items with value of zero usually contribute quite less or even worse nothing in the computation of documents relevancy. *Maximum Document Percentage* indicates the maximum number of documents where certain term can exist. For instance, if the *Maximum Document Percentage* is set to 50, then only the terms occur in less than 50 percent of all documents will be kept. The purpose of *Maximum Document Percentage* is to remove those terms that have quite high document frequency. If one term occurs in every document, then this term doesn't help to differentiate documents. By adjusting different values of *Minimum Document Frequency* and *Maximum Document Percentage*, the term vector space could be compressed in different degrees. TFIDF score is adopted as the term weight scheme. Here is given a definition of term vector in our project:

Definition *Suppose P = $\{p_1, p_2, ..., p_N\}$ is the web pages collection where N is the collection size, T = $\{t_1, t_2, ..., t_M\}$ is the term dictionary composed of all terms coming from the title, description and plain content of web page after stop list filtering and shrinking by Minimum Document Frequency. T is the dictionary size. Assign a TFIDF score $w_{i,j}$ for each vector item located by the pair (t_i, p_j). Then the term vector for each web page $\vec{p_j}$ = ($w_{1,j}$, $w_{2,j}$, ..., $w_{M,j}$), where j $\in$ (1, 2, ..., N).*

3.1.2 In-link Vector

Hyperlink is a special feature of web page, which implicates some relationship with other web pages. This kind of relationship seems to be structural, but can also be topical somewhat. For instance, if page *A* and page *B* share the same linkage to page *C* that is all about cat in topic, then page *A* and page *B* might also have the same topic of cat. It makes sense to consider both in-link vector and out-link vector for web page representation.

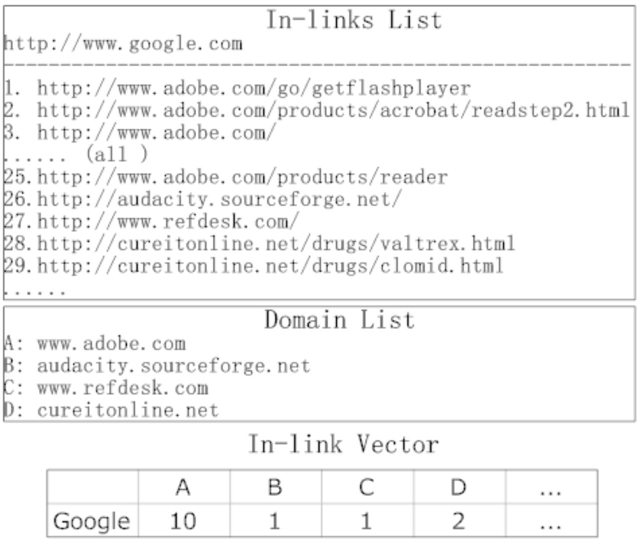
In general, the number of in-links of a certain page varies quite a lot, e.g. the web page *www.google.com* owns 249,792,548 in-links from all pages and 239,981,747 in-links except from the same domain retrieved by Yahoo! site explorer web service, but the page *http://www.languageguide.org/english/vocabulary/food/* owns only 16 in-links from all pages and 15 in-links except from the same domain. It is impossible to involve all in-links into the in-link vector if the number of in-links is very huge such as the web page *www.google.com* owns. Therefore, a pre-defined maximum of in-links for each web page is required since unlimited number of in-links will increase the dimension of in-link vector space dramatically. In our in-link vector construction, 100 is adopted as the maximum of in-links for each web page. The *Minimum Document Frequency* and *Maximum Document Percentage* factors applied for term vector construction are still workable for in-link and out-link vector construction.

It is very common that a web page owns many in-links from the same site, e.g. the web page *www.google.com* owns more than 20 in-links from the web site *www.adobe.com*. It is not very useful to contain all in-links from the same web site especially when the number of in-links from this web site is quite huge, because when adopting the count of in-links as the weight scheme, too big value of vector item will not contribute to the computation of in-link vector relevancy significantly, but increase the work load adversely. Therefore, we need to define the limitation of in-links from the same web site. A heuristic rule for differentiating weights among links is introduced in [19] and the detail description is presented in Equation 3.1, where *K* is the number of in-links from the same web site.

$$w_{i,j} = \begin{cases} 1 & \text{if K = 1} \\ 2 & \text{if } 1 < K < 20 \\ 3 & \text{if } K \geq 20 \end{cases} \tag{3.1}$$

For our in-link vector construction, we only adopt a pre-defined maximum of in-links from the same web site, e.g. 10. The another thing different from the in-link vector construction in [19] is that we use the domains of web pages instead of the URL addresses as the in-link vector items, and the 100 maximum of in-links per web page means the in-links from 100 distinct domains. Figure 3.1 demonstrates the in-link vector construction in our project. The web page *http://www.google.*

com has 25 in-links from the web site www. adobe. com, so the item of that domain is only weighted as 10.



In-links List
http://www.google.com

```
1.  http://www.adobe.com/go/getflashplayer
2.  http://www.adobe.com/products/acrobat/readstep2.html
3.  http://www.adobe.com/
......  (all )
25. http://www.adobe.com/products/reader
26. http://audacity.sourceforge.net/
27. http://www.refdesk.com/
28. http://cureitonline.net/drugs/valtrex.html
29. http://cureitonline.net/drugs/clomid.html
......
```

Domain List

```
A:  www.adobe.com
B:  audacity.sourceforge.net
C:  www.refdesk.com
D:  cureitonline.net
```

In-link Vector

	A	B	C	D	...
Google	10	1	1	2	...

Figure 3.1: An example of in-link vector construction.

Definition *Suppose P = $\{p_1, p_2, ..., p_N\}$ is the web pages collection where N is the collection size, D = $\{d_1, d_2, ..., d_K\}$ is the in-link domains collection corresponding to P where K is the collection size. Suppose S is the maximum of in-links from the same web site, X is the maximum of in-link domains for each page, then the in-link vector for each web page $\vec{inlink_j}$ = ($w_{1,j}$, $w_{2,j}$, ..., $w_{K,j}$), where j $\in$ (1, 2, ..., N), $w_{i,j}$ equals to the number of in-links from d_i and equals to S if the number of in-links from d_i is more than S. (In actual, TFIDF likewise weighting scheme is applied for in-link and out-link vector construction , which will be introduced detailedly in next section.)*

3.1.3 Out-link Vector

The out-link vector construction is quite similar to in-link vector construction. There are two kinds of out-links from the original web page: one is internal out-links that have the same domain as the original page, the other is external out-links that have different domains to the original page. A web page may contain only internal out-links, only external out-links or both of them. The internal out-links seems to contribute quite less to the computation of out-link vectors relevancy, but still have effects especially for the pages that only contain internal out-links. Therefore, we assign 1 to the vector item with the domain that all internal out-links belong to. If the page contains none of out-links, we will still assume it has at least one internal out-link. Similar to what we do for in-link vector construction, we also define a maximum of out-links from the same web site, e.g. 10. But we don't define the maximum of out-links for each page because the average number of out-links per web page is acceptable. Here is given a definition of out-link vector.

Definition *Suppose P = $\{p_1, p_2, ..., p_N\}$ is the web pages collection where N is the collection size, D = $\{d_1, d_2, ..., d_L\}$ is the out-link domains collection corresponding to P where L is the collection size. Suppose S is the maximum of in-links from the same web site, then the out-link vector for each web page $\vec{outlink_j}$ = ($w_{1,j}$, $w_{2,j}$, ..., $w_{L,j}$), where $j \in$ (1, 2, ..., N), $w_{i,j}$ equals to the number of out-links from d_i and equals to S if the number of out-links from d_i is more than S, but if d_i is the same domain as the page p_j no matter if the page p_j has internal out-links or not, the $w_{i,j}$ is set to 1.*

3.1.4 PageRank Vector

As described previously, Google PageRank is an integer value ranged from 0 to 10. The PageRank vector is constructed in two ways. The first way is to construct the PageRank vector with only one single value, e.g. 5. The second way is to construct the PageRank vector as a 11 dimensional vector, and assign 1 to the item located by the PageRank, e.g. the PageRank is 5 and thus only the 5th item of vector is set to 1, all rest items equal to 0. Here is given the definition of 11 dimensional PageRank vector.

Definition *Suppose P = $\{p_1, p_2, ..., p_N\}$ is the web pages collection where N is the collection size, Pr_j is the Goolge PageRank of page p_j. Then the PageRank vector for each web page $\vec{pagerank_j}$ = ($w_{0,j}$, $w_{2,j}$, ..., $w_{10,j}$), where $w_{Pr_j,j}$ equals to 1 and the rest are 0.*

3.2 Clustering System

Clustering system usually consists of documents crawling, indexing and clustering as its basic procedures. Our authoritative K-Means clustering method is implemented on top of Apache Lucene indexing, Apache Mahout vector creation and K-Means clustering components. Several other tools such as HtmlCleaner, HtmlParser are used for parsing web page to get content fragments and out-links, and Yahoo! site explorer is used to retrieve in-links of certain page.

3.2.1 Basic Principle

Figure 3.2 depicts the basic principle of our authoritative K-Means clustering procedure for web search results. After specifying certain queries, the Google search engine returns all matched results in a descendant order by its own ranking algorithm. Web page crawler downloads certain amount of search results to local storage for later content parsing and out-links extraction. Meanwhile, the in-links, out-links and Google PageRank for each search result are generated during web pages crawling. Manual check is performed since there might exist some error during the web page crawling, such as connection time out of certain web page, authorization of access to web page and etc. It is very common that not full percent of web search results are healthy to be clustered.

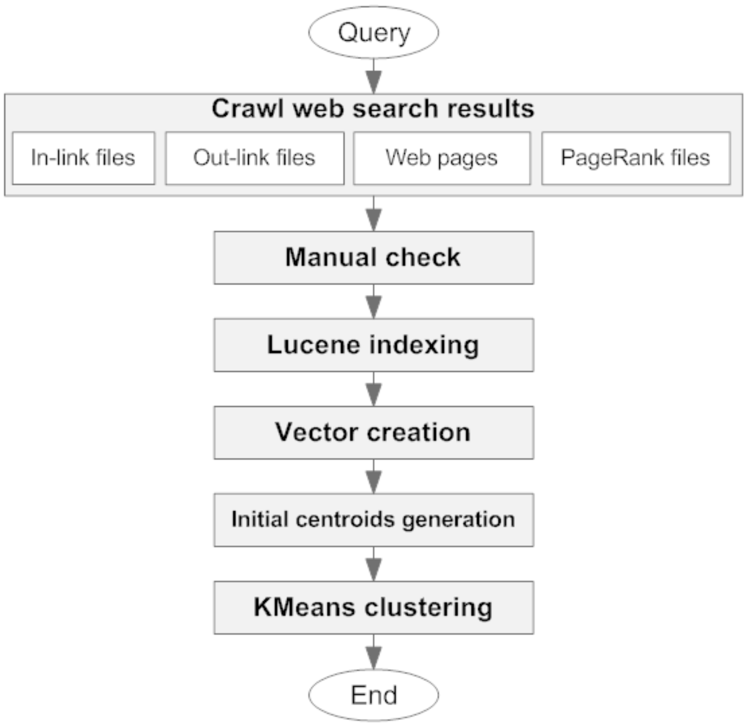


Figure 3.2: The basic principle of Authoritative K-Means clustering

When the crawling is completed, we use Lucene Indexing component to index the contents of web pages and their in-links and out-links information either. The index files are later converted into sequence files that are actually the vectors containers provided by Apache Hadoop. Through sequence file reader and writer, it is possible to operate on the vectors to either split a vector into several parts or merge two vectors into a new one. When the vectors are ready, we select pre-defined *K* cluster centroids randomly, which is prepared for K-Means clustering. The K-Means clustering assign each page to its closest cluster and re-compute each cluster centroid until a certain amount of iterations are finished or no cluster centroids changes.

3.2.2 Description of Components

As shown in Figure 3.2, there are several sub-procedures or components in our clustering system. They cooperate in a straight line pattern from one to another.

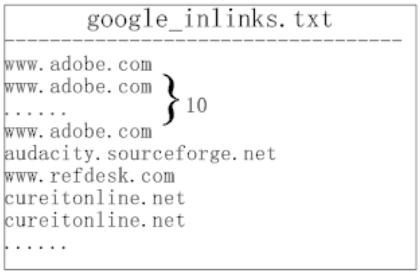
Web Search Results Crawler

Web search results crawler has four main responsibilities: download web search results, retrieve in-links, extract out-links and retrieve the PageRank for each search result. It simulates the web browser to send GET request containing the query information to Google search server and receives the matched results list as response. The query information usually includes query topic, the number of search results, starting position of search results and etc. Each search result page is downloaded for the further content parsing and out-links extraction.

Yahoo! site explorer in-link data web service is used for the in-links information retrieval. By accessing the web service request url address that contains some request parameters (e.g. the url address or domain of the page getting in-links, the number of in-links and etc.), the in-links list can be retrieved in an XML format. Through XML parser it is easy to retrieve all in-links. As described previously, the in-link vector is constructed with the distinct domains, therefore, only the domains of in-links need to be kept. We save all in-links domains for each search result page as a text file and treat the domains as terms. The biggest advantage of keeping in-links information in this way is that it is convenient for indexing and TFIDF weighting scheme can also be applied to in-link vector later. For instance, the in-link text file of search result page `http://www.google.com` in Figure 3.1 is finally looks like what is shown in Figure 3.3.

Through HTML parser, it is simple to extract the out-links of search result page. The out-links information is also kept as a text file just like the in-links text file. The PageRank is retrieved by accessing the url address of Google search page with the feature of PageRank. It is quite common that some web pages are not indexed and computed the PageRank yet by Google search server, in this case, it always assign 0 for the PageRank of these web pages. The PageRank of all web search results are stored in Java ArrayList object that is finally serialized to a *.ser* file, and is convenient to deserialize the *.ser* file into ArrayList object.

Manual Check



```
google_inlinks.txt
---------------------------------
www.adobe.com
www.adobe.com  }
......         } 10
www.adobe.com  }
audacity.sourceforge.net
www.refdesk.com
cureitonline.net
cureitonline.net
......
```

Figure 3.3: An example of in-links information text file.

During the web search results crawling, it is inevitable that some web pages have connection time out, some web pages have re-direction effects that are hard to be detected by crawler, some web pages have nested HTML page style and some web pages are just invalid to be open. All of these problems will cause the web search result page to be downloaded unsuccessfully. Therefore, the manual check of already downloaded web pages needs to be performed to remove those broken web pages. Because the in-links information is retrieved by Yahoo! site explorer inlink data web service, it might also cause the information retrieval failure, so the manual check of in-links is required as well.

Lucene Indexing
The main responsibility of Lucene Indexing component is to index web search results contents, in-links and out-links information. Different analyzers could be used to filter the web page contents, such as white space analyzer, stop list analyzer and etc. It is even possible to allow users to develop their own analyzers, e.g. stemming analyzer. Many meaningless terms will be removed by analyzers, and thus reduce the size of terms dictionary. Both HTML and text indexer are implemented to deal web pages and their in-links and out-links text files.

As described previously, the content of each web page is added as *Field* of *Document* into Lucene index file by *IndexWriter* of Lucene Indexing component. In our project, the concatenation of title, description and plain content of web page is added as the “content” *Field* into index file. Using the similar approach, both in-links information and out-links information are added as the “links” *Field* into index file. Later, the indexed information can be read out by *IndexReader*.

Figure 3.4 demonstrates the inner structure of Lucene *index* and the basic principle of indexing. The classes appear in Figure 3.4 are all main Lucene classes playing the core roles in indexing, which are listed below:

- **FSDirectory** is a straightforward implementation of Directory class as a di-

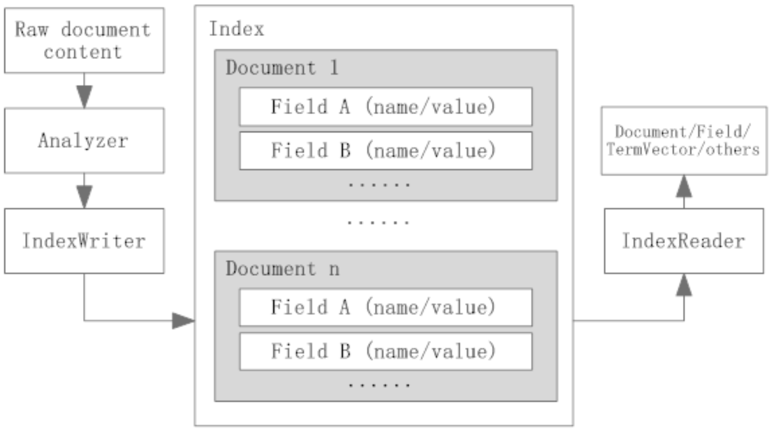


Figure 3.4: Basic principle of Lucene indexing.

rectory of index and its related files. It is always required for IndexWriter and IndexReader.

- **Field** is one section of a Document, such as *title*, *abstract* and etc. It is a pair of name and textual value, e.g. {“title”, “the cat is an animal”} is a Field with name of “title” and value of “the cat is an animal” respectively.
- **Document** is the unit of indexing and searching in Lucene. It is a set of fields. Fields are added to a document through the method *add* of Document class. When use the method *add*, it is possible to designate if the field needs to be indexed, stored with document in index file, or generate the term vector.
- **Analyzer** is an abstract class that provide the interfaces to analyze text. It thus represents a policy for extracting terms from text. Lucene provides several concrete analyzers derived from this abstract class, such as StopAnalyzer, StandardAnalyzer and etc.
- **IndexWriter** is the class that creates and maintains the index. Each document is added into the index file with method *addDocument*, updated with method *updateDocument* and removed with method *deleteDocument*. When all operations of IndexWriter are determined, the method *optimize* needs to be invoked to commit all operations, afterwards, IndexWriter needs to be closed with method *close*.
- **IndexReader** is an abstract class that provides an interface for accessing an index. Through the interface, it is possible to read related information of any Document, Field from the index. All concrete classes derived from IndexReader are usually constructed with a call of method *open*.

Vector Creation

In order to create vectors that can be used for clustering, it needs to convert Lucene index files into Hadoop sequence files first. The sequence file which can be seen as a container of vectors. In our project, we use Mahout to implement the vector creation. Although the vector creation is much more based on Apache Hadoop since the core class *SequenceFile* belongs to Hadoop and some classes such as *IndexReader* and etc from Lucene are also involved, we implement our vector creation based on a *Driver* program developed by Mahout.

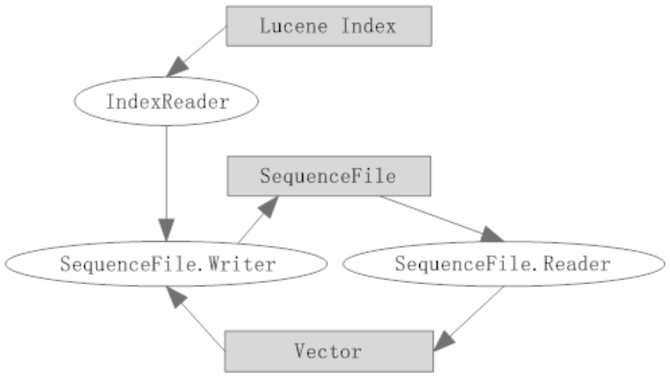


Figure 3.5: The basic procedure of vector creation from Lucene index.

Figure 3.5 depicts the basic procedure of vector creation from Lucene index. After Lucene indexing, different kinds of information of all web pages, such as content of pages exist as certain *Field* of *Document* in index file. These information can be read out by *IndexReader* from index file, and written into *SequenceFile* by *SequenceFile.Writer*. *SequenceFile* is a class belonging to Hadoop. It is flat file consisting of binary Key/Value pairs, and provides *Writer*, *Reader* and *Sorter* classes for writing, reading and sorting of sequence files respectively. For the generation of sequence file, the normalization and different weight schemes are possible to be applied, such as TF, TFIDF score and etc. *SequenceFile.Reader* resolves the sequence file into a number of vectors, and *SequenceFile.Writer* takes either *Vector* or Lucene index as input to generate sequence file. The *SequenceFile* is one input of Mahout K-Means clustering.

The vector creation component is also used for the vectors combination. Our authoritative K-Means clustering will be performed for different vectors combination to study the effects of different types of authoritative information. Different vectors are read out from their own sequence files respectively and are merged into a new vector or even more complicated a new vector combining different vectors with different weights. Finally, the new vector is written back into sequence file.

Initial Centroids Generation

When all vectors are ready for K-Means clustering, it still needs to determine the initial clusters centroids. We use the class *RandomSeedGenerator* under *org.apache.mahout.clustering.kmeans* package of Mahout to select certain amount of initial centroids from the input data collection (the collection of vector representations of web search search results) randomly. It is possible to change the number of cluster centroids K by such a methods, e.g. $K = 12$. Once the K initial cluster centroids are determined, the final clusters number is also equal to K (this is decided by the internal implementations of class *RandomSeedGenerator* and *KMeansDriver*).

K-Means Clustering

With the initial clusters generated in "Initial Centroids Generation" procedure and the vector representations of pages, K-Means clustering can be performed with certain amount of iteration. Figure 3.6 describes several core classes of Mahout used for our clustering. *RandomSeedGenerator* is used for the initial cluster centroids generation (it determines the K number of K-Means clustering). *KMeansDriver* then takes the initial centroids and the sequence file of vector representation of pages as input to run the classic K-Means clustering.

It is noticed that several distance measures are available for clustering, such as *EuclideanDistanceMeasure*, *CosineDistanceMeasure* and etc. All of these distance measure classes implement the interface *DistanceMeasure*. *CosineDistanceMeasure* is mostly used in our project, but we also implement our own distance measure class *WeightedCosineDistanceMeasure* that can support weighted cosine distance measure of multi-vectors. In our project, we use the *CosineDistanceMeasure* and *WeightedCosineDistanceMeasure* for K-Means clustering.

3.3 Adjustment of Authoritative K-Means Clustering

We use K-Means clustering implementations of Mahout for our authoritative K-Means clustering, whose principle is briefly depicts in Figure 3.7. The K-Means clustering of Mahout has already been specified and it is impossible to change its internal implementation. But as shown in Figure 3.7, the clustering algorithm takes vectors, K initial centroids and certain distance measure as input, which gives us the opportunity to adjust this clustering algorithm for our authoritative K-Means clustering by changing the vectors and developing our own distance measures.

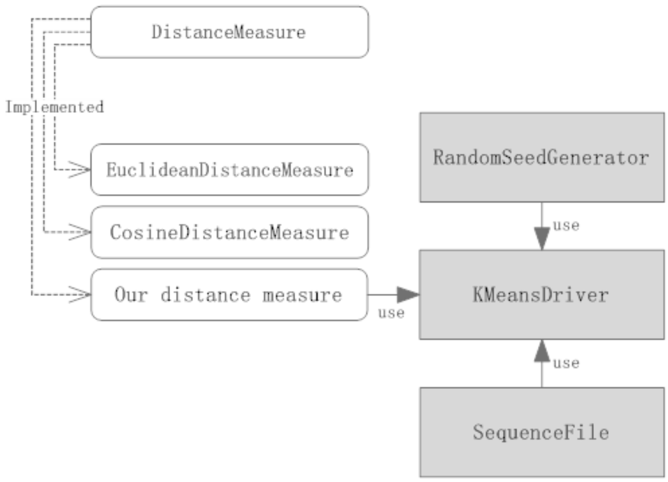


Figure 3.6: Mahout K-Means clustering.

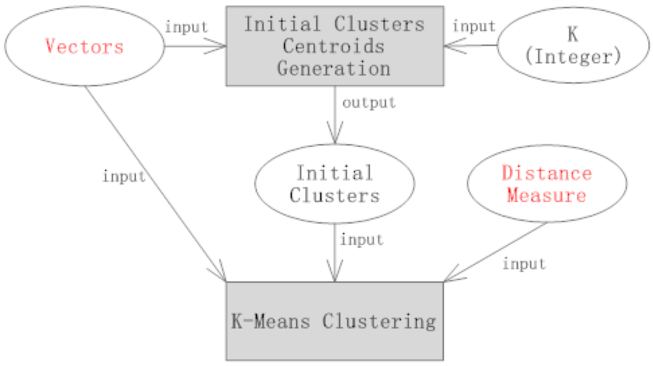


Figure 3.7: Authoritative K-Means clustering.

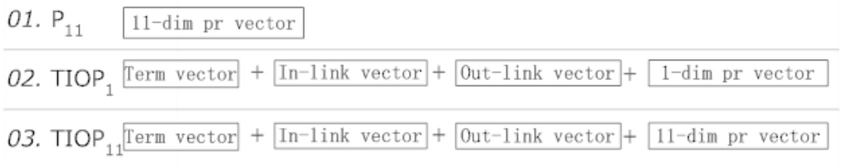
3.3.1 Vectors Combination

Term vector, in-link vector, out-link vector and PageRank vector are four primary elements to represent web page. By combining any types of vector equally or assigned with different weights, we can construct a new vector to represent web page, and thus study the effects of different vector types trying to figure out the most optimized vectors combination.

In our project, we design a couple of combinations involving all four types of vector above, and name them in a special rule can make them more understandable during the later experiments. The rule is described as:

- Make the capital letter *T* as the abbreviation of term vector, *I* for in-link vector, *O* for out-link vector and *P* for PageRank vector.
- There are two types of PageRank vector, one is 1-dimensional and the other is 11-dimensional, we use P_1 and P_{11} to express them respectively.
- Vectors are combined equally and the effect of order is ignored. E.g. *TIO* means the combination of term vector, in-link vector and out-link vector, and it is the same to no matter *TOI* or *OIT*. TIP_{11} indicates the combination of term vector, in-link vector and 11-dimensional PageRank vector.

We group the types of vectors combination into 1-vector combination, 2-vector combination, 3-vector combination and 4-vector combination, which are depicted in Figure 3.8, Figure 3.9 and Figure 3.10.

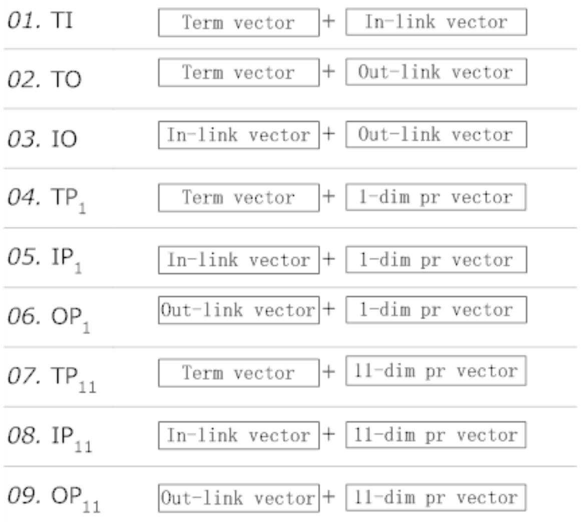


01. P_{11} 11-dim pr vector

02. $TIOP_1$ Term vector + In-link vector + Out-link vector + 1-dim pr vector

03. $TIOP_{11}$ Term vector + In-link vector + Out-link vector + 11-dim pr vector

Figure 3.8: Vector combination with 1 or 4 vectors.



01. TI Term vector + In-link vector

02. TO Term vector + Out-link vector

03. IO In-link vector + Out-link vector

04. TP_1 Term vector + 1-dim pr vector

05. IP_1 In-link vector + 1-dim pr vector

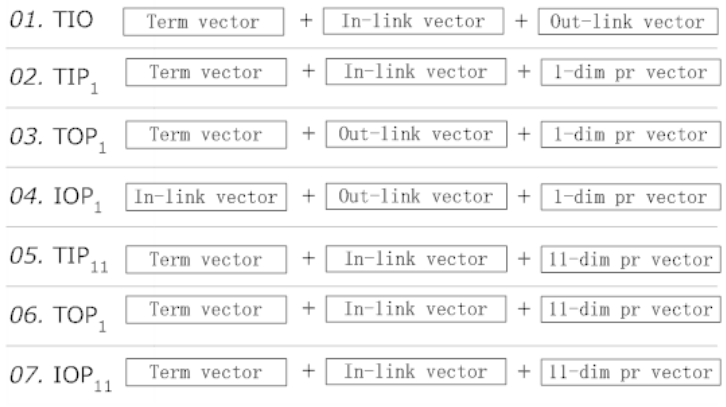
06. OP_1 Out-link vector + 1-dim pr vector

07. TP_{11} Term vector + 11-dim pr vector

08. IP_{11} In-link vector + 11-dim pr vector

09. OP_{11} Out-link vector + 11-dim pr vector

Figure 3.9: Vector combination with 2 vectors.



01. TIO	Term vector	+	In-link vector	+	Out-link vector
02. TIP_1	Term vector	+	In-link vector	+	1-dim pr vector
03. TOP_1	Term vector	+	Out-link vector	+	1-dim pr vector
04. IOP_1	In-link vector	+	Out-link vector	+	1-dim pr vector
05. TIP_{11}	Term vector	+	In-link vector	+	11-dim pr vector
06. TOP_1	Term vector	+	In-link vector	+	11-dim pr vector
07. IOP_{11}	Term vector	+	In-link vector	+	11-dim pr vector

Figure 3.10: Vector combination with 3 vectors.

3.3.2 Weighted Cosine Distance Measure

Another thing we can adjust for our clustering method is the distance measure. All vector combinations listed above have equal effect of each element vector, and it is generally sensible to differentiate the effects of different element vector, e.g. out-link vector has been studied to have less effect than term vector or in-link vector [19][20]. Therefore, it is sensible to assign different weight for each element vector.

Cosine distance measure is adopted as the computation of web pages relevancy for our clustering, one advantage of cosine distance measure is that it takes normalization into account. We develop a new distance measure called *Weighted-CosineDistanceMeasure* by utilizing both vector combination and cosine distance measure to simulate the vector combination with different weights of element vectors. Figure 3.11 shows an example of weighted vectors combination to illustrate the basic principle of weighted cosine distance measure. In this example, each vector is actually composed of term vector and in-link vector, normal cosine distance measure is applied on term vectors and in-link vectors respectively, and finally their cosine distance measure results are integrated with different weights.

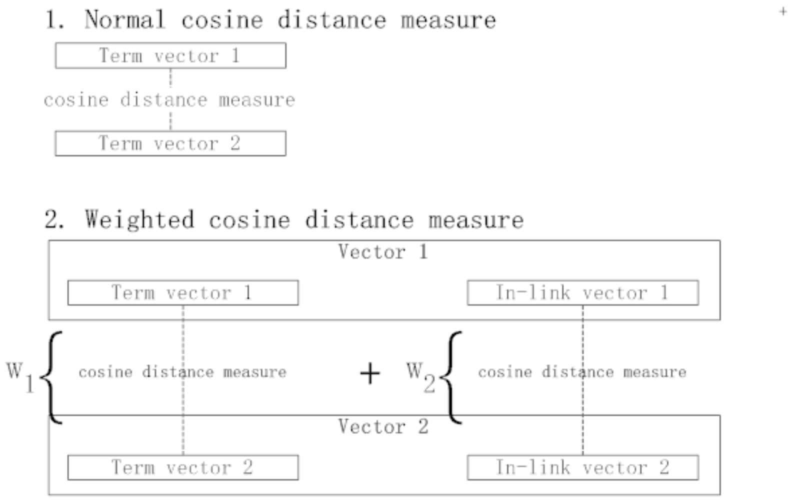


Figure 3.11: An example of weighted vectors combination.

Chapter 4

Experiments and Evaluation

We perform several experiments to compare different clustering methods, study the effects of different elements of page representation on clustering, such as content, in-link, out-link and PageRank. In addition, we design two types of PageRank vectors: 1-dimensional and 11-dimensional to study the difference on clustering as well. In this chapter, we first introduce the data collections and the test system configuration. Second, we describe the experiments design in detail to reveal how these experiments are organized and performed. Third, a short description of what kind of clustering evaluation metrics we use is covered. Finally, we present the results and evaluation.

4.1 Data Collection

By learning from [19], we select two query topics like “jaguar” and “big apple” to retrieve the web search results from Google search engine to be our data collections. For each topic, we select top 200 pages and extract 100 as the maximum of in-link domains for each page (some page will have only a few in-links). The in-links are obtained by Yahoo! site explorer web service. The total number of out-link domains for each page is not limited. Besides, we decide the maximum number of in-links and out-links from the same domain to be 10. During the web search results retrieval, some web pages are not healthy with the possible problems of encryption, redirection, invalid and etc, therefore, not full percent of 200 pages for each topic will be further clustered.

As mentioned previously, there is one parameter called *Minimum Document Frequency* that can help reduce the dictionary size (vector dimension). During the vector creation procedure that convert Lucene index to sequence file, we set up the *Minimum Document Frequency* to be 3, 2 and 1 for content, in-link and out-link

Table 4.1: Data collections and their classes.

Topic	Size	number of classes	class description	Max class	Min class
jaguar	194	13	jaguar car, animal, sports, travel, business, shopping, education, entertainment, military, society, music, IT, miscellaneous	114	1
big apple	196	14	New York, news, education, business, shopping, travel, entertainment, media, art, personal, animal, sports, food, miscellaneous,	39	1

respectively. The determination of these *Minimum Document Frequency* parameters is based on the tests of data collections to keep the whole vector dimension (combine content, in-link and out-link) with a reasonable size.

We perform our initial evaluations by manually checking all retrieved pages for each topic, to create some classes. Each class corresponds to a sub-topic under the general query topic, such as “sports”, “animal” under query topic “jaguar”. Although this process is time-consuming and might lead to bias in the final evaluations, it is still necessary because the pre-determined classes are required for our evaluation metrics, e.g. entropy, precision and recall. Table 4.1 shows the data collections and their classes information.

4.2 Experiment Environment Configuration

Some primary hardware and software configuration details of the experiments system are list as below:

- **CPU**: Pentium(R) Dual-Core T4200 2.00 GHz
- **Memory**: Hynix DDR3 SDRAM 1066(533 MHz) 2GB
- **Operating System**: Windows 7 Ultimate 32-bit
- **JRE**: 1.6.9_20 (Version 6 Update 20)

4.3 Experiments Setup

We perform three experiments correspondingly to study the difference among different clustering methods, the difference of 1-dimensional and 11-dimensional PageRank vectors and the effect of different element vectors in clustering. As shown in Table 4.1, the data collection of topic "jaguar" contains 194 pages spreading in 13 manually created classes and the data collection of topic "big apple" contains 196 pages spreading in 14 classes. By considering the data collection size and the manual classes creation, we arbitrarily determine the number *K* in K-Means clustering to be 12 for all three experiments. By providing different vector combinations and distance measures, we are able to set up these three experiments differently. We will use some abbreviations for different clustering methods to be remembered easily. All abbreviations are listed as follows:

- P_1: 1-dimensional PageRank vector.
- P_{11}: 11-dimensional PageRank vector.
- T: Content-based clustering.
- IO: Link-based clustering, which combine both in-links and out-links.
- TIO: Content-link coupled clustering.
- TP_1: Clustering based on the combination of content and P_1.
- TP_{11}: Clustering based on the combination of content and P_{11}.
- IOP_1: Link and P_1 combined clustering.
- $TIOP_1$: Clustering based on the combination of content, links and P_1.

4.3.1 Experiment for Comparison of PageRank Vector Types

We provides two types of PageRank vector with 1-dimensional and 11-dimensional respectively. This experiment is trying to study on the effects of different types of PageRank vector on clustering. To reduce the influence by other types of vector as much as possible, e.g. in-link vector and out-link vector, we only use TP_1 and TP_{11}. Both two data collections are used for the experiment. In this experiment, we weight term vector and PageRank vector equally. The specification of this experiment is shown in Table 4.2.

Table 4.2: Specification of experiment "Comparison of Different PageRank vectors".

K (clusters number)	12
Data collection	jaguar, big apple
Vector combination	TP_1, TP_{11}
Distance measure	CosineDistanceMeasure

Table 4.3: Specification of experiment "Comparison of Different Clustering Methods".

K (clusters number)	12
Data collection	jaguar, big apple
Vector combination	T, $I^{0.5}O^{0.5}$, $T^{0.5}I^{0.3}O^{0.2}$, $I^{0.5}O^{0.3}{P_1}^{0.2}$, $T^{0.5}I^{0.3}O^{0.1}{P_1}^{0.1}$,
Distance measure	WeightedCosineDistanceMeasure

4.3.2 Experiment for Comparison of Different Clusterings

The purpose of first experiment is to study the difference among different clustering methods: content-based, link-based, link-PageRank combined, content-link combined and content-link-PageRank combined clustering. Both two data collections of query topic "jaguar" and "big apple" are used for this experiment. According to some papers [19][24] that already study on the effect of content, in-link and out-link, we don't use the vector combination with each type of vector weighted equally. For instance, term vector usually weights more than the other types of vector. Besides, both two types of PageRank vector are adopted for corresponding vector combinations. Once the *K* initial cluster centroids is determined, all clustering methods share it. The specification of this experiment is shown in Table 4.3.

The superscript number of each vector type indicates its weight, e.g. $I^{0.5}O^{0.5}$ indicates in-link vector weights 50% and out-link vector weights another 50%. The subscript number of PageRank vector indicates the different dimensions.

4.3.3 Experiment for Effect of Different Element Vectors

The third experiment is to study on the effect of different types of vector in the vector combination. Although some paper [19][24] have done the similar study, but because we adjust the in-link and out-link vector construction (described in "Web Page Representation" section of previous chapter) and involve the PageRank vec-

Table 4.4: Specification of experiment “Effect of Different Types of Vector”.

K (clusters number)	12
Data collection	jaguar
Vector combination	TIOP$_1$
Distance measure	WeightedCosineDistanceMeasure

tor, we decide to make this study again.

In this experiment, only one data collection with topic “jaguar” and TIOP$_1$ are used. We set up a series of weights combinations for TIOP$_1$ shown in Table 4.5. We prefer to set a distinct weight for each type of vector rather than to only focus on the weights combination of only one or two vector types. The specification of this experiment is shown in Table 4.4.

4.4 Results and Evaluation

Before we present our experiment results and evaluation, we would like to introduce two evaluation metrics we use: Entropy, Precision and Recall. Afterwards, detailed experiment results and evaluation will be discussed.

4.4.1 Evaluation Metrics

Entropy [35] is an external quality measure that enables us to evaluate how well the clustering is working by comparing the groups generated by the clustering methods to known classes. It is frequently applied for K-Means clustering [36][37]. Low entropy indicates high purity of the cluster due to the high intra-cohesiveness while high entropy indicates that the pages within the same cluster are not related highly, which means that the pages with different topics are clustered together. Entropy can be computed by the combination of Equation 4.1, 4.2 and 4.3:

$$E = -\sum_{i=1}^{K} \frac{m_i}{m} E_j \tag{4.1}$$

$$E_j = \sum_{i=1}^{L} p_{ij} \log p_{ij} \tag{4.2}$$

Table 4.5: Series of weights combination for different vector types.

No	**T**	**I**	**O**	**P_1**
1	0	0.333	0.333	0.333
2	0.5	0.25	0.125	0.125
3	0.5	0.125	0.125	0.25
4	0.125	0.125	0.125	0.625
5	0.125	0.125	0.625	0.125
6	0.125	0.25	0.125	0.5
7	0.125	0.25	0.5	0.125
8	0.125	0.375	0.125	0.375
9	0.125	0.5	0.125	0.25
10	0.125	0.625	0.125	0.125
11	0.25	0.125	0.125	0.5
12	0.25	0.125	0.5	0.125
13	0.25	0.25	0.125	0.375
14	0.25	0.25	0.25	0.25
15	0.25	0.25	0.375	0.125
16	0.25	0.375	0.125	0.25
17	0.25	0.5	0.125	0.125
18	0.333	0.333	0	0.333
19	0.333	0.333	0.333	0
20	0.375	0.125	0.125	0.375
21	0.375	0.125	0.25	0.25
22	0.375	0.125	0.375	0.125
23	0.375	0.375	0.125	0.125
24	0.625	0.125	0.175	0.125

$$p_{ij} = \frac{m_{ij}}{m_j} \tag{4.3}$$

Where p_{ij} is the probability that a page of cluster *j* belongs to class *i*; m_j is the number of pages in cluster *j* and m_{ij} is the number of pages of class *i* in cluster *j*; *L* and *K* are the number of classes and clusters respectively; E_j is the entropy of cluster *j* and *E* is the total entropy of clustering method.

Precision and Recall are re-defined in the context of classification or clustering. Clustering can be alternatively interpreted as a series of decisions, one for each of the $N(N-1)/2$ pairs of documents (*N* is collection size) in the collection [1]. The computation of precision and recall are shown in the Equation 4.4 and 4.5 respectively, where *TP* is a true positive decision that assigns two similar pages to the same cluster, *FP* is a false positive decision that assigns two dissimilar pages to the same cluster, *FN* is a false negative decision that assigns two similar documents to different clusters and there is also a true negative decision *TN* that assigns two dissimilar pages to different clusters. Recall is also referred to as the True Positive Rate [38]. A good clustering usually has higher precision and recall value.

$$Precision = \frac{TP}{TP+FP} \tag{4.4}$$

$$Recall = \frac{TP}{TP+FN} \tag{4.5}$$

4.4.2 Comparison of 1-Dimenasionl and 11-Dimensional PageRank Vector

We run different clusterings on the data collection with topic "jaguar" and "big apple" based on TP_1 and TP_{11} respectively. According to the experiment results, we evaluate their qualities on the metrics shown in Figure 4.1, 4.2 and 4.3.

In Figure 4.1, it shows that the TP_1 has lower entropy then TP_{11}, which means the clusters obtained by TP_1 have higher purity, many pages with the same topics are clustered into the same group. Since Google PageRank is a good authoritative information of web page that implies the links relationship among web pages, it indicates the similarity of web page somewhat, e.g. the page owning PageRank 7 might be similar to a page owning PageRank 6 with more probability than a page owning PageRank 0. P_1 strengthens this kind of similarity more than P_{11}, which is illustrated in Figure 4.4. For P_{11}, two pages with PageRank 6 and 7 are not relevant in PageRank part, but for P_1, these two pages are similar somewhat. This

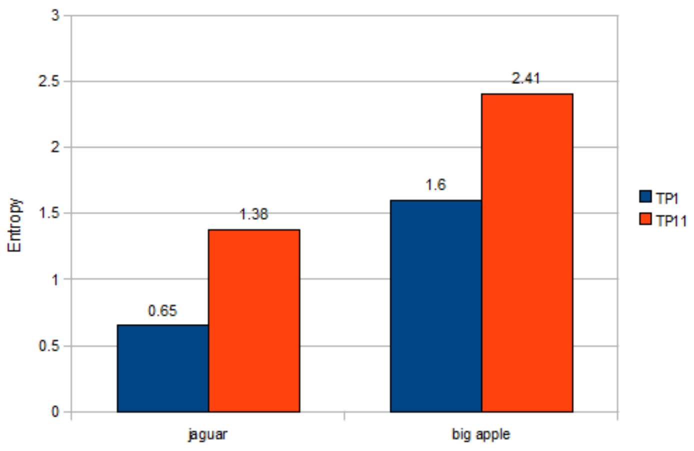


Figure 4.1: Comparison of different PageRank vectors based on entropy.

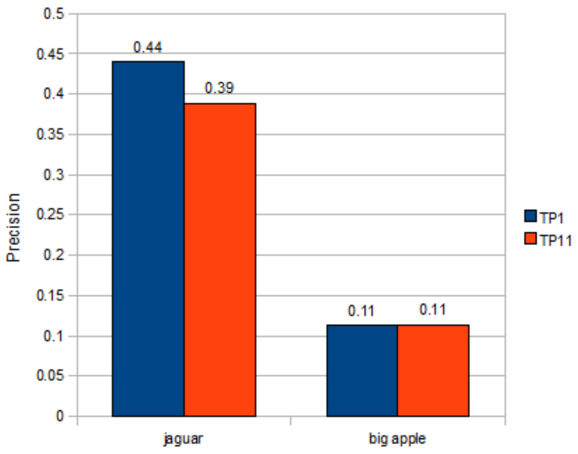


Figure 4.2: Comparison of different PageRank vectors based on precision.

effect is also reflected in Figure 4.3. Recall is referred to as the rate of true positive decision that assigns two similar pages into the same group, and TP$_1$ has higher recall value than TP$_{11}$. In addition, TP$_1$ also shows a better precision on the data collection with topic "jaguar" than TP$_{11}$.

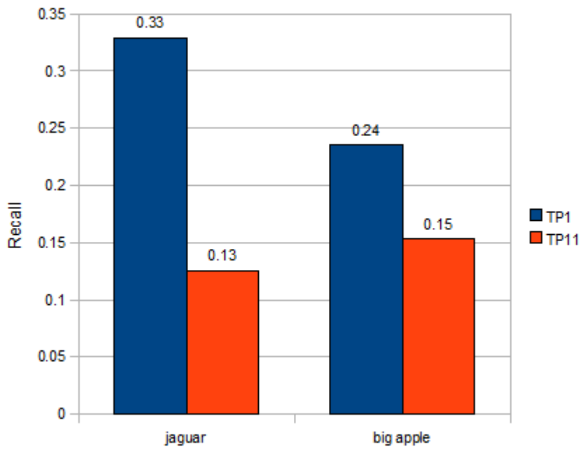


Figure 4.3: Comparison of different PageRank vectors based on recall.

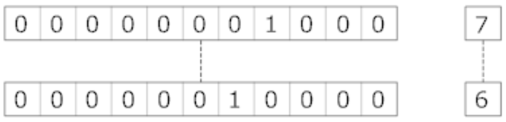


Figure 4.4: Difference of similarity by P_{11} and P_1.

4.4.3 Comparison of Different Clustering Methods

After adjusting the construction of both in-link and out-link vectors and involving a new PageRank vector, we need to compare the different clusterings based on different page representations, such as content-based, link-based, link-PageRank combined, content-link coupled and content-link-PageRank combined clustering. Because P_1 shows better effect than P_{11}, we only consider P_1 in this experiment. The evaluation by entropy, precision and recall are shown in Figure 4.5, 4.6 and 4.7 correspondingly.

As shown in Figure 4.5, T clustering has the lowest entropy and thus the best clusters quality. TIOP$_1$ is worse than T clustering but still the best among all rest four clustering methods. One possible reason why TIOP$_1$ is worse than T clustering is the weight of content. Content is the main feature of documents, and the content weight in TIOP$_1$ is only 0.5, but 1 in T clustering. If we look at the TIO and TIOP$_1$ that have the same content weight, TIOP$_1$ performs better than TIO, which means P_1 does help improving the clustering. Compare IO and IOP$_1$ on data collection with topic “jaguar” that have the same in-link weight, P_1 also shows its effect on improving the clustering.

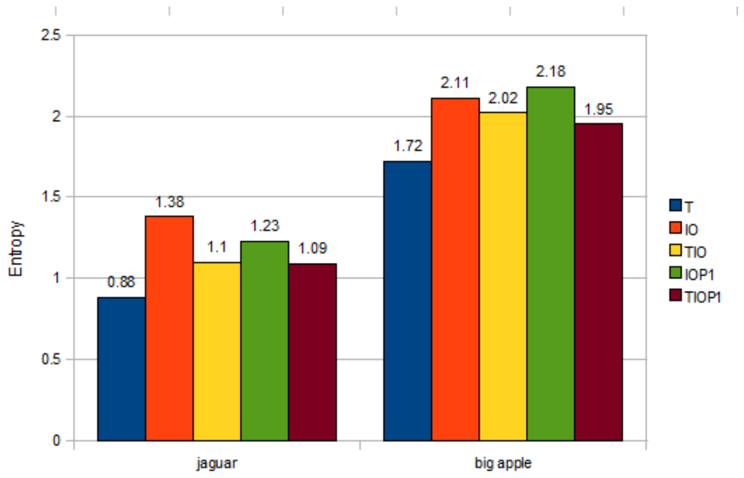


Figure 4.5: Comparison of different clusterings based on entropy.

4.4.4 Effect of Term, In-link, Out-link and PageRank

As discussed previously, on data collection with topic “jaguar”, TIOP$_1$ shows its goodness, e.g. TIOP$_1$ has second lowest entropy and the highest precision and recall. Therefore, we still believe TIOP$_1$ can produce better clustering results. In this part, we evaluate different weights of content, in-link, out-link and PageRank trying to figure out an optimized weights assignment.

Figure 4.8 and 4.9 show the effect of different elements on clusters distribution. Out-link affects the size of maximum cluster and the number of small clusters most over the other elements. Too big cluster will usually collect many noisy pages that reduce the cluster purity, and too many small clusters will increase the possibility of scattering similar pages to different clusters. Therefore, the weight of out-links can not be kept small. It also shows the PageRank has the least effect on the size of maximum cluster and in-link has the least effect on the number of small clusters.

By calculating the average evaluation metrics value of different weights of a single element, we are able to estimate the effect of different element on clustering clearly. Figure 4.10, 4.11, 4.12 and 4.13 show the average evaluation for content, in-link, out-link and PageRank respectively. All trend lines in figure are of polynomial with order 2.

As shown in Figure 4.10, as weight of content increases, the entropy of clustering is reduced, which means it generates clusters with more purity. According its trend line, the lowest value might be reached when the weight is between 0.6 and

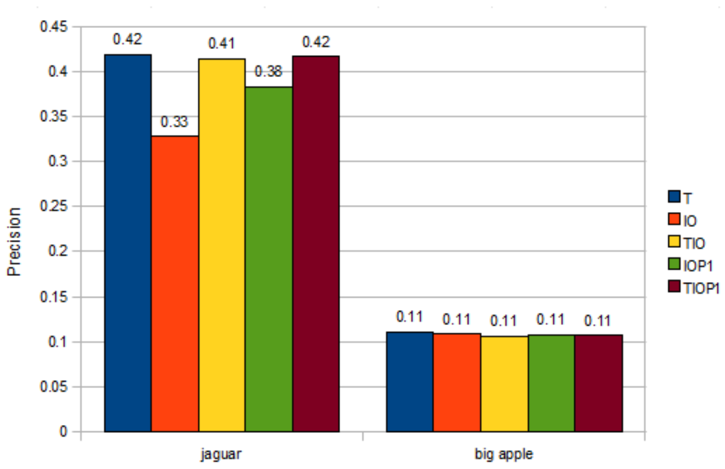


Figure 4.6: Comparison of different clusterings based on precision.

In Figure 4.6, T, TIO and TIOP$_1$ have the highest precision, which indicates the importance of content in clustering. Link-based clustering such as IO and IOP$_1$ shows insufficient in clustering on data collection with topic "jaguar". The evaluation precision, recall of clusterings on data collections with topic "big apple" shows some different results compared to the data collection with topic "jaguar". In Figure 4.6, all clustering methods show almost the same but quite low precision, and in Figure 4.7, IO and IOP$_1$ even perform better over T, TIO and TIOP$_1$. This might be caused by the bias of manual determination of classes on data collection with topic "big apple", or the data collection has much more overlaps on both in-links and out-links.

0.7. The peak of both precision and recall seems to be reached when the weight of content locates between 0.35 and 0.4, but both of their trend lines are quite flat indicating the difference between different weights are not significant.

Similar to content, there is no big difference of precision and recall on in-link. But the entropy is decreased when the weight increases. The lowest value is reached when the weight of in-link is between 0.1 and 0.15 as shown in Figure 4.11

Out-link has the smallest influence on entropy, precision and recall as shown in Figure 4.12, which means that out-link is not a factor affecting the clustering result significantly.

The entropy on PageRank reaches a lowest value when its weight is between 0.6 and 0.65 in our experiment as shown in Figure4.13. This weight range is close to the one of content. Especially the lowest entropy of all weight assignments in our experiment is the one with weights 0.125, 0.125, 0.125, 0.625 for content, in-

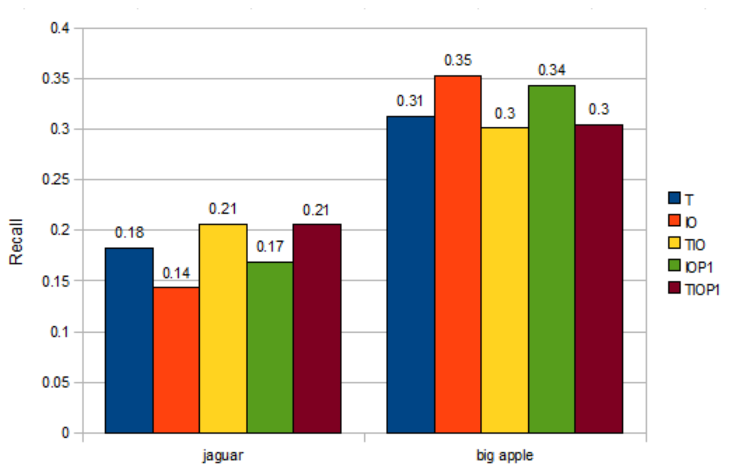


Figure 4.7: Comparison of different clusterings based on recall.

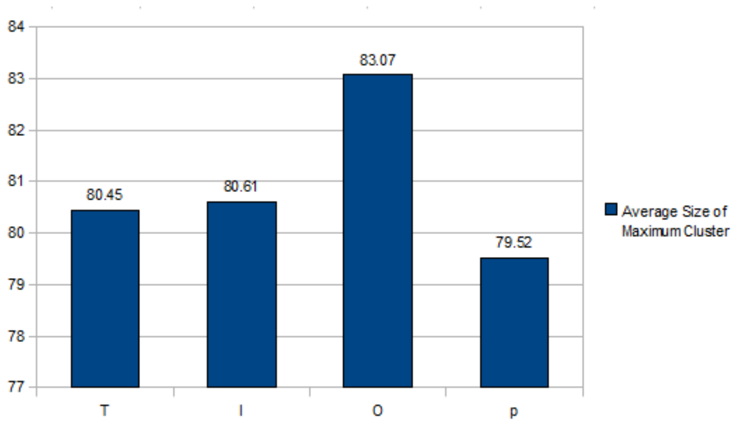


Figure 4.8: Comparison of average size of maximum cluster.

link, out-Link and PageRank respectively. It might because we use only one singe number of PageRank, which has only 11 distinct numbers, and therefore can generate much more similarity.

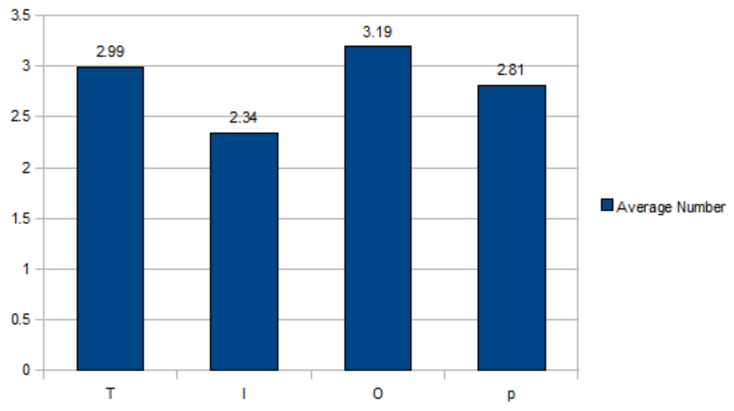


Figure 4.9: Comparison of average number of small cluster (size ≥ 5).

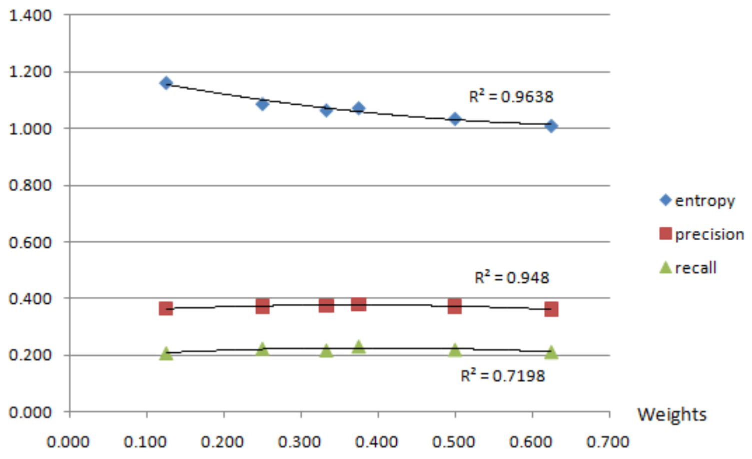


Figure 4.10: Average evaluation of content with different weight assignments.

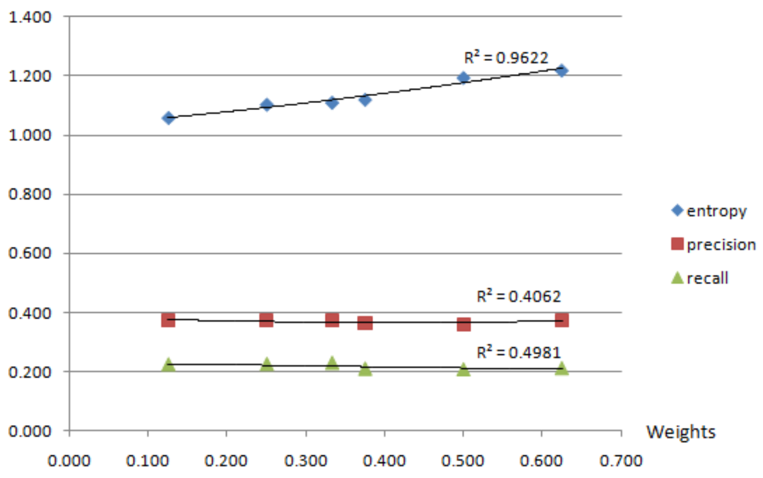


Figure 4.11: Average evaluation of in-link with different weight assignments.

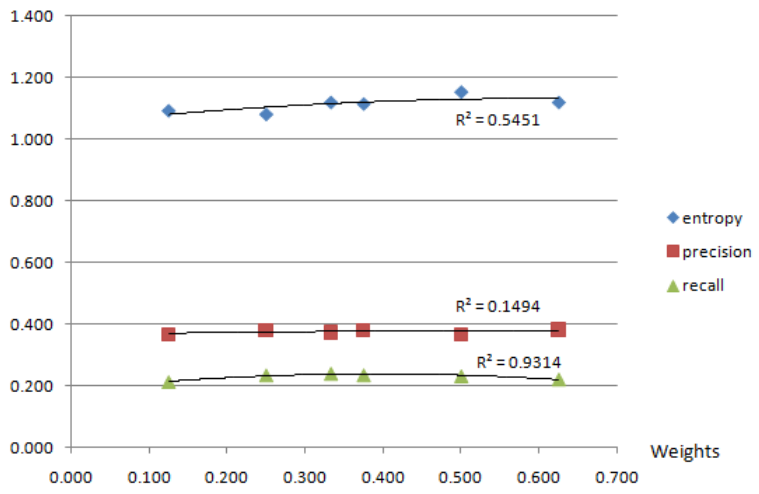


Figure 4.12: Average evaluation of out-link with different weight assignments.

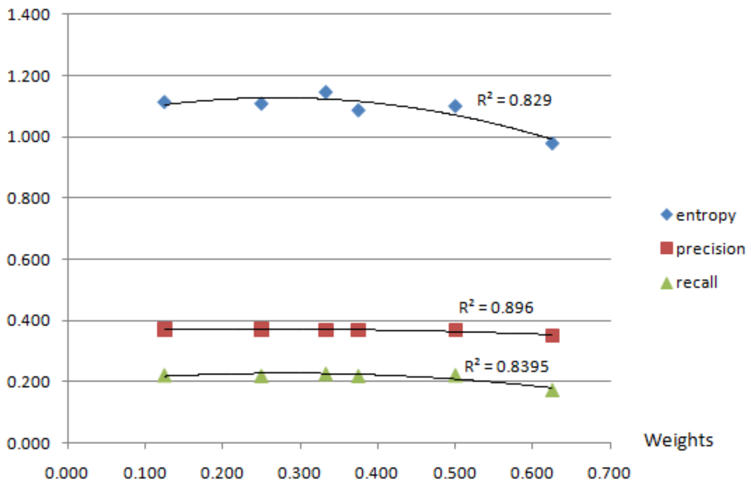


Figure 4.13: Average evaluation of PageRank with different weight assignments.

Chapter 5

Conclusion and Future work

In this chapter, we will give a summary of what we study and get through our authoritative K-Means clustering and look into some other knowledge or techniques that we can use to improve our authoritative K-Means clustering in the future.

5.1 Conclusion

We propose a new authoritative K-Means clustering method for web search results, which combines content, in-link, out-link and Google PageRank. In order to increase the overlap among in-links and out-links of web pages, we adjust the construction of in-link and out-link vector by using the domains of hyper links, not their concrete URLs. Two types of PageRank vector are introduced into our study, one is a single value representation of PageRank and the other is 11-dimensional vector of PageRank. Since the constructions of in-link and out-link vector are changed and a new PageRank vector is involved, it is necessary to study their effects on clustering. We design and perform three different experiments to study the difference of two types of PageRank vector, to compare the clustering methods based on different combinations of content, in-link, out-link and PageRank, and finally to evaluate the effects of all these four elements on clustering for web search results.

According to he experimental results and evaluations, Google PageRank does help improve the clustering for web search results, and further a single value representation of PageRank performs better than 11-dimensional PageRank vector because the former one can bring more possible similarity. Among all clustering methods we study in our project, the clustering based on coupled content-link and PageRank shows its competitiveness: it is the best in both precision and recall evaluation on clustering results and the second best in entropy in our experiments.

Out-link has the most effect on the size of maximum cluster and the number of small clusters whose size is less than 5, which means out-link might make similar pages to be scattered to different groups and cluster dissimilar pages into the same group. Content has the biggest effect on clustering, the weight ranged from 0.6 to 0.65 for content reaches the peak effect of content on clustering. In-link and out-link don't impact the clustering significantly under the new construction of in-link and out-link vector, the weights of in-link from 0.1 to 0.15 reaches a better entropy value in our experiments, and due some negative effects of out-link on clustering, it is better to assign the weight less than 0.1 for out-link. The PageRank reaches its best entropy when it is assigned a weight from 0.6 to 0.65, which is the highest value range that we give to PageRank. It's effect on clustering seems to be the same to content, but since the PageRank only has 11 distinct values, and if we assign a bigger weight to PageRank, it will bring much more similarity among documents and this is what we think a possible reason why PageRank has a low entropy when its weight is big.

5.2 Future Work

Although the PageRank shows some goodness on clustering for web search result, but the effect is still need to be studied deeper to determine the optimized weight range. The PageRank we use in our project is actually a general PageRank, therefore, the query topic sensitive vector can be first considered to improve the relevancy of PageRank and its web page since web search results to be clustered are also query topic related. The second thing that we can do in the future is to combine other clustering algorithms with K-Means to increase the quality of clustering results, such as hierarchical clustering and etc. More data collections with bigger size can be applied for more clusterings. In addition, we don't consider the weights of different part of content, such as title, description, anchor text and plain content, but they actually will have different effects on representing the web page content, therefore, the weighted parts of web page content can be studied for clustering.

Bibliography

[1] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[2] Wikipedia, "Information retrieval." [online], accessed 22.04.10, http://en.wikipedia.org/wiki/Information_retrieval.

[3] Wikipedia, "Vector space model." [online], accessed 22.04.10, http://en.wikipedia.org/wiki/Vector_space_model.

[4] T. C. for Neuroinformatics, "Vector space model." [online], accessed 22.04.10, http://isp.imm.dtu.dk/thor/projects/multimedia/textmining/node5.html.

[5] Wikipedia, "Stemming." [online], accessed 11.05.10, http://en.wikipedia.org/wiki/Stemming.

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

[7] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web: Experiments and models," in *Proceedings of the Ninth Conference on World Wide Web*, (Amsterdam, Netherlands), pp. 309–320, ACM Press, May 2000.

[8] R. Angelova and S. Siersdorfer, "A neighborhood-based approach for clustering of linked document collections," in *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, (New York, NY, USA), pp. 778–779, ACM, 2006.

[9] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, no. 1-7, pp. 107–117, 1998.

[10] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[11] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[12] M. Henzinger, “Hyperlink analysis on the world wide web,” in *HYPERTEXT ’05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, (New York, NY, USA), pp. 1–3, ACM, 2005.

[13] T. H. Haveliwala, “Topic-sensitive pagerank,” in *Eleventh International World Wide Web Conference (WWW 2002)*, 2002.

[14] X. Zhang, X. Hu, and X. Zhou, “A comparative evaluation of different link types on enhancing document clustering,” in *SIGIR ’08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, NY, USA), pp. 555–562, ACM, 2008.

[15] Y. Wang and M. Kitsuregawa, “Link based clustering of web search results,” in *WAIM ’01: Proceedings of the Second International Conference on Advances in Web-Age Information Management*, (London, UK), pp. 225–236, Springer-Verlag, 2001.

[16] Y. Wang and M. Kitsuregawa, “Use link-based clustering to improve web search results,” in *WISE ’01: Proceedings of the Second International Conference on Web Information Systems Engineering (WISE’01) Volume 1*, (Washington, DC, USA), p. 115, IEEE Computer Society, 2001.

[17] Y. Cai, P. Li, H. Liu, J. He, and X. Du, “S-simrank: Combining content and link information to cluster papers effectively and efficiently,” in *ADMA ’08: Proceedings of the 4th international conference on Advanced Data Mining and Applications*, (Berlin, Heidelberg), pp. 317–329, Springer-Verlag, 2008.

[18] R. Angelova and S. Siersdorfer, “A neighborhood-based approach for clustering of linked document collections,” in *CIKM ’06: Proceedings of the 15th ACM international conference on Information and knowledge management*, (New York, NY, USA), pp. 778–779, ACM, 2006.

[19] Y. Wang and M. Kitsuregawa, “Evaluating contents-link coupled web page clustering for web search results,” in *CIKM ’02: Proceedings of the eleventh international conference on Information and knowledge management*, (New York, NY, USA), pp. 499–506, ACM, 2002.

[20] D. S. Modha and W. S. Spangler, “Clustering hypertext with applications to web searching,” in *HYPERTEXT ’00: Proceedings of the eleventh ACM on Hypertext and hypermedia*, (New York, NY, USA), pp. 143–152, ACM, 2000.

[21] X. He, H. Zha, C. H.Q. Ding, and H. D. Simon, “Web document clustering using hyperlink structures,” *Computational Statistics & Data Analysis*, vol. 41, pp. 19–45, November 2002.

[22] D. Koller and M. Sahami, “Hierarchically classifying documents using very few words,” in *ICML ’97: Proceedings of the Fourteenth International Conference on Machine Learning*, (San Francisco, CA, USA), pp. 170–178, Morgan Kaufmann Publishers Inc., 1997.

[23] "Hierarchical clustering." [online], accessed 27.05.10, http://www.resample.com/xlminer/help/HClst/HClst_intro.htm.

[24] Y. Wang and M. Kitsuregawa, "Enhancing contents-link coupled web page clustering and its evaluation," 2004.

[25] K. Avrachenkov, V. Dobrynin, D. Nemirovsky, S. K. Pham, and E. Smirnova, "Pagerank based clustering of hypertext document collections," in *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, NY, USA), pp. 873–874, ACM, 2008.

[26] Wikipedia, "Jensen¨cshannon divergence." [online], accessed 27.05.10, http://en.wikipedia.org/wiki/Jensen\discretionary{-}{}{}Shannon_divergence.

[27] Apache, "Lucene." [online], accessed 24.05.10, http://lucene.apache.org/.

[28] E. Hatcher and O. Gospodnetic, *Lucene in Action (In Action series)*. Greenwich, CT, USA: Manning Publications Co., 2004.

[29] Apache, "Hadoop." [online], accessed 05.06.10, http://hadoop.apache.org/common/.

[30] Apache, "Mahout." [online], accessed 02.06.10, http://mahout.apache.org/.

[31] G. Ingersoll, "Introducing apache mahout: Scalable, commercial-friendly machine learning for building intelligent applications," 2009.

[32] SourceForge.Net, "Htmlcleaner." [online], accessed 25.05.10, http://htmlcleaner.sourceforge.net/index.php.

[33] SourceForge.Net, "Htmlparser." [online], accessed 25.05.10, http://htmlparser.sourceforge.net/.

[34] Yahoo, "Yahoo site explorer web service." [online], accessed 02.06.10, http://developer.yahoo.com/search/siteexplorer/.

[35] C. E. Shannon, "A mathematical theory of communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, no. 1, pp. 3–55, 2001.

[36] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *KDD-2000 Workshop on Text Mining, August 20* (M. Grobelnik, D. Mladenic, and N. Milic-Frayling, eds.), (Boston, MA), pp. 109–111, 2000.

[37] Y. Zhao and G. Karypis, "Criterion functions for document clustering: Experiments and analysis," tech. rep., 2002.

[38] Wikipedia, "Precision and recall." [online], accessed 27.05.10, http://en.wikipedia.org/wiki/Precision_and_recall.