Optimization of Internet Search based on Noun Phrases and Clustering Techniques

R. Subhashini
Research Scholar, Sathyabama University,
Chennai-119, India

V. Jawahar Senthil Kumar
Assistant Professor, Anna University,
Chennai, India

ABSTRACT
Information Retrieval plays a vital role in our daily activities and its most prominent role marked in search engines. Retrieval of the relevant natural language text document is of more challenge. Typically, search engines are low precision in response to a query, retrieving lots of useless web pages, and missing some other important ones. In this paper, we present linguistic phenomena of NLP using shallow parsing and Chunking to extract the Noun Phrases. These noun phrases are used as key phrases to rank the documents (typically a list of titles and snippets returned by a certain Web search engine). Organizing Web search results in to clusters facilitates user's quick browsing through search results. Traditional clustering techniques are inadequate since they don't generate clusters with highly readable names. Here, we also proposed an approach for web search results clustering based on a phrase based clustering algorithm Known as Optimized Snippet Flat Clustering (OSFC). It is an alternative to a single ordered result of search engines. This approach presents a list of clusters to the user. Experimental results verify our method's feasibility and effectiveness.

Keywords
Noun Phrases; Document Clustering; Information Retrieval; Natural Language Processing; Web Mining

1. INTRODUCTION
The World Wide Web is a very large distributed digital information space. The ability to search and retrieve information from the Web efficiently and effectively is an enabling technology for realizing its full potential. Information Retrieval (IR) plays an important role in search engines. Today's most advanced search engines use the keyword-based ("bag of words") paradigm, which concedes some inherent disadvantages. Typically, search engines [1] are low precision in response to a query, retrieving lots of useless web pages, and missing some other important ones.

The task of finding relevant information [6] from the web using just few words is a tremendous challenge. However, search engines [10] do not organize documents automatically; the problem of categorizing a large source of information into groups of similar topics is still unsolved. Document clustering [9] is widely applicable in areas such as search engines, web mining, information retrieval, and topological analysis. Moreover, most traditional clustering algorithms [4] cannot be directly used for search result clustering, because of some practical issues. The research on clustering search results [2] has been investigated in a number of previous works. Agglomerative Hierarchical Clustering (AHC) algorithms are probably the most commonly used. These algorithms are typically slow when applied to large document collections.

The VSD model [14] uses a feature vector to represent a document. Most of the document clustering methods is based on VSM that represents documents as the feature vector of the terms. Phrase searching is a very efficient way to achieve the desired result than performing a keyword search. STD model considers document as a sequence of words and extract all overlap phrases in the document. This method is performed by phrase based analysis i.e., the similarity between the documents should be based on matching phrases rather than single words only. The Motivation of this thesis is to enhance the end users need to find meaningful results from the web search engine results. This is achieved by automatically categorizing the search results and by presenting the clustered output result to the user. Two different techniques (VSD & STD) have been integrated to enhance the user's ability to access search results.

Suffix Tree Clustering [5] produces too many base clusters. Because of its high dimensionality it is not suitable for large document collections. The proposed phrase clustering is modified from the Suffix Tree Clustering by further calculating several important properties to identify the score of the phrases and base clusters before merging the similar clusters. The new document clustering algorithm we proposed is to improve the quality in clustering web page snippets, and the clustering speed can meet the demand of "on the fly" mode. An NLP-based IR system [7] has the goal of providing more precise, complete information in response to a user's real information need. In this paper, we explore shallow NLP techniques to support a range of NL queries [8] and web snippets over an existing keyword-based engine. Given a query and the ranked list of search results, our method first parses the whole list of snippets, and also the query to extract all possible noun phrases using shallow NLP Chunking from the contents, and calculates the ranking of the snippets using cosine similarity measures. Then these ranked snippets are clustered using the phrase based clustering algorithm. In this algorithm, several properties for each phrase such as phrase frequencies, document frequencies, phrase length, etc are considered and score of the phrase is calculated. The phrases are ranked according to the score, and the top-ranked phrases are taken as base clusters, which are further merged according to their corresponding documents. Our method (OSFC) is more suitable for Web search results clustering because we emphasize the efficiency of identifying relevant clusters for Web users. Furthermore, the clusters are ranked according to their scores, thus the more likely clusters required by users are ranked higher. It generates shorter and

more readable cluster names, which enable users to quickly identify the topics of a specified cluster.

2. SYSTEM OVERVIEW

In this framework, Natural Language Processing tasks, Information Retrieval System (IRS) and phrase based document clustering are combined together. The architecture diagram is shown in figure 1. The Natural Language Processing tasks [3] are used to retrieve the noun phrases from the input query and also the snippets collected from search engines. Then these noun phrases are compared with the query using TFIDF algorithm and cosine similarity measures in Vector Space Model (VSM). Cosine similarity measures are used to rank the retrieved documents based on the scores of the similarity measures. These ranked documents are then used to create the clusters and is presented to the user for easy access instead of searching a long list of documents.

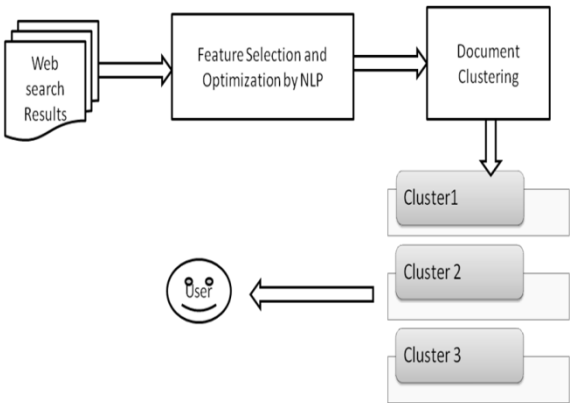

Fig 1: System Architecture

Our system is active at the beginning and end of the search session and we follow a typical Service Oriented Architecture, with a consumer (our system) and provider (Yahoo!). Consequently, we are able to partition the functionality in two disparate phases.

2.1 Phase One

In Phase One, variants of the input query are optimized and submitted using the API calls, and each retrieves up to 100 results. Then the snippets are optimized to extract the noun phrases using shallow NLP [15]. We must then perform a re-ranking on the results as a whole, based on some relevance measure.

2.1.1 Noun Phrase Extractor

Information retrieval systems of all kinds rely on base noun phrases as their primary source of entity identification. Because this is such a necessary task to natural language processing, there are a multitude of algorithms designed to handle it. A noun phrase is a syntactic unit of the sentence in which information about the noun is gathered. The noun phrase extractor and is made up of three main modules: tokenization; part-of-speech tagging; noun phrase identification using Chunking. Before implementing these modules, the input snippets should be split in to separate sentences using the Split method on this input will result in an array with five elements, when we really want an array with only two. This split method is mainly used to detect the end of sentences.

2.1.1.1 Tokenization

Tokenization process is used to determine sentence boundaries, and to separate the text into a stream of individual tokens (words) by removing extraneous punctuation. It separates the text into words by using spaces, line breaks, and other word terminators in the English language. Document texts must be tokenized correctly in order for the noun phrase extractor to parse the text efficiently. It also is a basis for a phrase boundary detection.

2.1.1.2 Part-of-speech tagging

A part-of-speech tagger assigns a part of speech label to each word in a text depending on the labels assigned to the preceding words. Often, more than one part-of-speech tag is assigned to a single word, presumably reflecting some kind of ambiguity in the input. It's task is to assign a syntactic category to each word in a text, thereby resolving some ambiguities. Following the completion of the tagging process, the noun phrases will be identified. Having obtained an array of tokens from the tokenization process, we can feed that array to the part-of-speech tagger. The POS tags are returned in an array of the same length as the tokens array, where the tag at each index of the array matches the token found at the same index in the tokens array.

2.1.1.3 Chunking and shallow parsing

Chunking and Shallow Parsing aim at separating words in a sentence into basic phrases, e.g. noun phrases or simple verb phrases. It is mainly used to find phrases. It could be useful when looking for units of meaning in a sentence larger than individual words. Phrase chunking is the task of segmenting a text into information units larger than a word and possibly comprising other chunks, e.g. 'the black board' forms a noun phrase. Chunking recognizes the boundaries of noun phrases based on POS. Text chunking divides the input text into such phrases and assigns a type such as NP for noun phrase, VP for verb phrase, PP for prepositional phrase in the following example, where the chunk borders are indicated by square brackets:

[NP I] [VP ate] [NP the noodles] [PP with] [NP sauce]

2.1.2 Information Retrieval System (IRS)

In IRS, the noun phrases extracted from chunker are given as input and this is compared with the user query. Comparison is also similar to that of Vector space model (VSM) i.e., instead of key phrases, noun phrases are used as indexes of the VSM. Term frequencies and inverse document frequencies are calculated using the following formula. TFIDF [17] is one popular technology in information retrieval and exploration. TFIDF is a statistical method to evaluate the importance of one word or term or phases in one of the files of document sets or knowledge warehouse. The traditional TFIDF algorithm, by Gerald Salton, is a method to describe the features of document for vector space information retrieval paradigm. The inverse document frequency IDFi of a phrase ti can be calculated as follows :

$$IDF_i = Log_{10}\left(\frac{N}{n_i}\right) \quad (1)$$

where IDFi denotes the inverse document frequency of phrase ti; N denotes the number of documents(snippets) in the document set; ni denotes the number of documents containing term ti. The weight wik of phrase ti in snippet dk can be calculated as follows:

$$W_{ik} = \frac{tf_{ik}}{max_j\, tf_{jk}} \times IDF_i \quad (2)$$

where tfik denotes the frequency of phrase ti in snippet dk and max tfjk denotes the maximal frequency of phrase in snippet dk. From which weight of the noun phrases are calculated. After that Similarity Measures are calculated between these weighted noun phrases and queries using the cosine similarity measures. Cosine similarity is one of the most popular similarity measure applied to text documents, such as in numerous information retrieval applications and clustering too. For a document D and query Q, the cosine similarity measure is given by:

$$\mathrm{Sim(D,Q)} = \frac{\sum_i (d_i * q_i)}{\sqrt{\sum_i d_i^2 * \sum_i q_i^2}} \quad (3)$$

Where d, q are m-dimensional vectors over the term set T = {t1, . . . , tm}. Each dimension represents a term with its weight in the document, which is non-negative. Based on the scores of the similarity measures the snippets are ranked.

2.2 Phase Two

Clustering techniques play an important role in searching and organization of web pages. In this phase, the ranked documents obtained as output is given as an input the OSFC clustering algorithm. This algorithm is used to generate the clusters to the users based on the shared phrases and is also used to re rank the clusters based on the score of the clusters.

2.2.1 Optimized Snippet Flat Clustering (OSFC)

This method has five logical steps: (1) Search result fetching, (2) Document pre processing, (3) Building the suffix tree and selection of base cluster, (4) Merging base clusters, (5) Scoring final clusters

2.2.1.1 Search result fetching

The first step is to get the ranked snippets of search results returned by a certain Web search engine from Phase One. These snippets are analyzed and result items are extracted. We assume these contents are informative enough because most search engines are well designed to facilitate user's relevance judgment only snippet, thus it is able to present the most relevant contents for a given query. Each extracted phrase is in fact the name of a candidate cluster, which corresponds to a set of documents that contain the phrase.

2.2.1.2 Pre processing

Pre-processing is selecting the most suitable terms that describe better content. The terms are transformed using stemming algorithm. Non-word tokens, such as articles, pronouns, prepositions, etc., are eliminated [16]. The snippets are given to stop word removal module. As mentioned in the "document cleaning", stop words are removed from the given data. Extracted snippet information is given to OSFC module for insertion. The suffix tree data structure has been introduced as an efficient string processing technique. A suffix tree allows us to insert a string into the suffix tree incrementally. The given snippets are split as mentioned in the OSFC algorithm and formed Suffix tree along with cluster information that also will be updated. Chim et al. [11] also proposed a new definition "stop node", which applies the same idea of stop words in the suffix tree similarity measure computation. A node with a high document frequency df can be ignored in the similarity measure, but the corresponding phrase and its words might be kept by other nodes in the suffix tree. In this paper, we adopt the same idea of "stop node" instead of stop words, which also prove to be efficient.

2.2.1.3 Selection of base clusters

Once the suffix tree formation is completed, evaluate the phrase importance by statistical method which is usually applied in VSD model. In this paper, the equation for computing an interesting score, shown in Eq.7, is modified from Hung Chim et.al [11].

In Zamir's STC the score of the base cluster is calculated by:

$$S(B) = |B| \times f(|P|) \quad (4)$$

Where $|B|$ is the number of documents in B, and $|P|$ is the number of words in P. Then all base clusters are sorted by the scores, and the top k base clusters are selected for cluster merging. In this the properties of the phrases are not considered, hence it produces more number of base clusters.

According to Hung Chim et.al [11], all the nodes n of the common suffix tree are mapped to a M dimensional space of VSD model (n = 1,2……..M), each document d can be represented as a feature vector of the weights of M nodes, d = {w(1,d),w(2,d),.....w(M, d)}

When we represent each document as a feature vector in the M dimensional space, it's very easy to understand that the document frequency of each node df(n) is the number of the different documents that have traversed node n; the term frequency tf(n; d) of a node n with respect to a document d is the total traversed times of document d through node n. The suffix tree constructed from documents usually contains lots of internal nodes (phrases). Not all internal nodes (phrases) are useful for document clustering, and some of the nodes (phrases) may give irrelevant results. So selecting a subset of original nodes as document features can reduce the high dimensionality of the feature space and also improve the accuracy of clustering results. For each internal node ni, the phrase designated by ni is pi. When we are constructing the suffix tree, use variable tf (pi) accumulate the times traverse through the node ni by the suffix in the documents corpus, then tf(pi) is the term frequency of phrase pi ; the times of different documents that traverses through the node ni is df(pi) , then df(pi) is the document frequency. Therefore the weight of phrase pi in documents corpus i.e., the weight w(n, d) of node n in document d can be calculated using the classic tf / idf scheme in formula (5), where N is the total number of documents in corpus.

$$tf, idf = \log\big(tf(p_i)\big) * \log\left(\frac{N}{df(p_i)}\right) \quad (5)$$

These phrase frequency and inverse document frequency properties are supposed to be relative to the salience score of phrases. Intuitively, more frequent phrases are more likely to be better candidates of salient phrases; while phrases with higher document frequency might be less informative to represent a distinct topic.

A phrase is an ordered sequence of one or more words. The more number of words a phrase contains, then richer meaning it can express. Therefore, the importance of a phrase should incorporate a factor about the length of phrase pi, which designated by │pi │.The phrase length property is simply the count of words in a phrase. For example, LEN ("Computer") =1and LEN ("Information Retrieval Systems") =3. Generally, a longer name is preferred for users' browsing. We calculate the factor using a heuristic utility function in following formula:

$$f(|p_i|) = log_2|p_i| \quad (6)$$

The time cost for finding all nodes traversed by a suffix tree is decided by the length of its longest common prefix (LCP) in the suffix tree. The maximum length of LCPs in the suffix tree of standard documents is only 4 in our experimental data sets.

The score s(ni) of node ni (phrase pi) is given by formula(7).

$$S(n_i) = \sum(tf, idf) * f(|p_i|) * |d| \quad (7)$$

Where │d│is the number of snippets in cluster ni. Then the base clusters containing the highest scoring phrases i.e., top n ranked base clusters should be selected for merging. The initial time complexity of cluster merging process is O(n2).To keep this cost as constant, the similarity is not calculated for all base clusters but only for top 'n' ranked base clusters. Here, the value of n= 500 was sufficient to ensure better performance.

2.2.1.4 Merging base Clusters

Zamir's STC [5] defines a binary similarity measure between base clusters based on the overlap of their document sets. This measure is used to merge base clusters. Given two base clusters Bm and Bn, with sizes │Bm│ and │Bn│ respectively, and representing the number of documents common to both base clusters. The similarity of Bm and Bn to be 1 if:

$$|B_m \cap B_n|/|B_m| > 0.5 \quad \text{and}$$

$$|B_m \cap B_n|/|B_n| > 0.5 \quad (8)$$

Otherwise, their similarity is defined to be 0.

It is found that the "and" Boolean operator is not suitable in the following condition if one base cluster is the subset of the other i.e., Bm C Bn. So the "and" operator is changed to "or" operator:

$$|B_m \cap B_n|/|B_m| > \propto \quad \text{or}$$

$$|B_m \cap B_n|/|B_n| > \propto \quad (9)$$

This is essentially identical to Yang's [12] improvement. Here the α varies from 0.4 to 0.9 and it shows better performance at α =0.6. A smaller α can result in a large number of merged clusters, but the similarity between the documents in the merged cluster is not strong. A bigger α can generate very similar documents in the same merged cluster but the number of the merged clusters is usually less. Since it uses the top 'n' ranked base clusters for merging, only less number of clusters are formed than the existing method.

2.2.1.5 Scoring final clusters

The score of the final cluster is calculated by

$$S(C) = \sum_{n_i \in C} S(n_i) \quad (10)$$

Where ni is the base cluster belonging to final cluster C.

The salience phrases are then ranked by the score. Then based on the scores of these base clusters, the final clusters are ranked. After salient phrases are ranked, the corresponding document lists constitute the candidate clusters, with the salient phrases being cluster names. Meanwhile, the cluster names are adjusted according to the new generated cluster i.e., any label or combination of labels in the merged cluster should be a good general description of the documents in the cluster. However, it is noted that some labels are the subsets of one another, which may be merged into the same clusters. We do not want to double count them. Labels created are based on these shared phrases and are more easily readable and understandable to the user. More specially, we choose the longest label first, and then choose the labels which are not the subsets of any selected labels. Finally, the topmost clusters are shown to user. When a user selects a cluster, the corresponding document list is shown to the user.

3. EXPERIMENTAL SETUP

3.1 Evaluation Measures

Performance measures of Information retrieval are precision and recall. Precision (P) is the fraction of retrieved documents that are relevant.

$$Precision\ (P) = \text{Relevant Items Retrieved / Retrieved Items} \quad (11)$$

$$\text{Recall (R)} = \text{Relevant Items Retrieved/Relevant Items} \quad (12)$$

3.2 Dataset Preparation

Due to lacks of standard dataset for testing web search results clustering, we have to build a small test dataset. For this purpose, we have defined a set of queries, for which search results were collected from the search engine. We manually assigned a relevance judgment (relevant or not) to each document in these collections.

Web documents dataset are collected with the help of Yahoo API. Yahoo! Search BOSS [13] (Build your Own Search Service) is an initiative in Yahoo! Search to open up Yahoo!'s search infrastructure. This release includes Web, News, and Image Search as well as Spelling Suggestions. Developers have expressed interest in altering the result order, removing results they do not want, and blending in their own data. All of these activities are allowed and encouraged with BOSS but not in the existing search API. This API returns the XML, that dump the search results and it contains the URL, Title, Snippet, and Description.

3.3 Results

The table 1. shows the various 10 queries used to generate the web document datasets and the corresponding clusters generated for each query by using the proposed phrase clustering algorithm.

Table 1. Queries Used For Evaluation

Query	No. of clusters
Anatomy of Computer Architecture	7
Why Computer Security is important	6

Advantages of Database Management System over File System	8
Uses of Internet	9
Various Programming Languages	5
Importance of Networking in an Organization	6
Advantages of Firewall Software	4
Data mining and Data warehousing	7
Multimedia Applications in Mobile Wireless	6
What is cloud computing	8

The precision values are calculated for various recall points and is shown in figure 2.

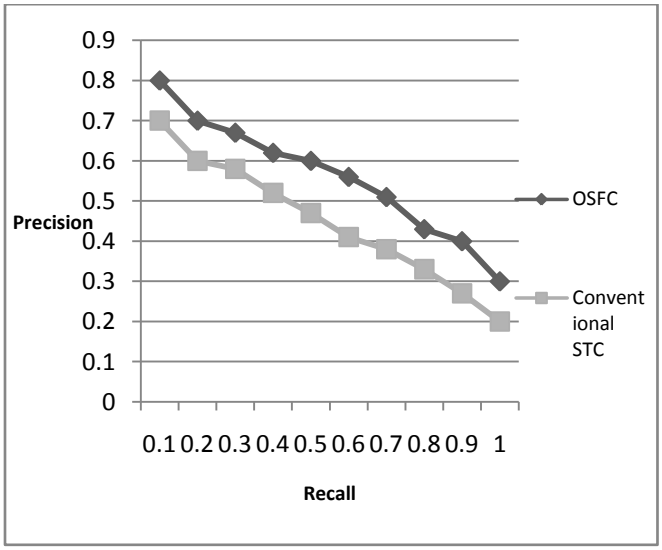

Fig 2: Precision/Recall Curve

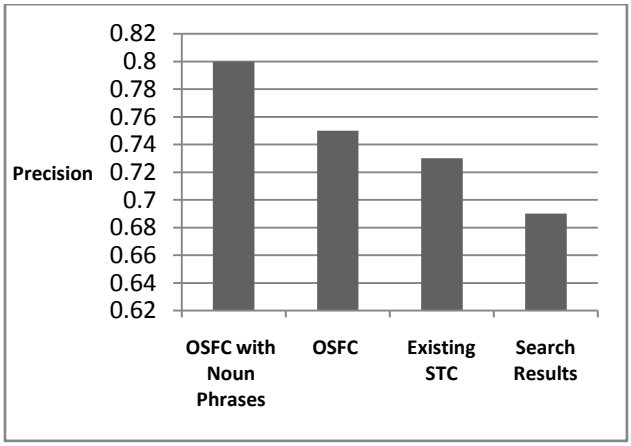

Fig 3: The Average Precision

The figure 3 illustrates the comparison of precision values between the proposed system and the existing systems with VSM, OSFC with and without noun phrases. The above figure shows that the OSFC with noun phrases has achieved an increment of 11% on the average precision over Search Results and 7% over the existing STC. This suggests that NL queries may be used to improve the accuracy of web search, through shallow NLP systems and the new phrase clustering algorithm. This improvement is mainly due to two reasons, i.e., we extract only the noun phrases from the snippets, and remove the meaningless phrases which misguide the clustering result and selection of the base clusters for merging.

4. CONCLUSION AND FUTURE ENHANCEMENTS

Natural Language Processing (NLP) techniques have been used in Information Retrieval mainly for presenting the user the documents that best satisfy their information needs. This paper suggests that Shallow NLP techniques are used to improve the accuracy of web search. In Shallow NLP the Chunker is used to extract the noun phrases. Then these noun phrases are used to retrieve the relevant documents. Hence it improves the performance than using the key phrases. From the results it can be assumed that NL search systems are useful to the society, besides the NL paradigm can be used to improve the precision of web search. As a synchronous wrapper around Yahoo! the proposed system is inherently slower than Yahoo! In this work we have shown a novel document clustering algorithm (OSFC) by modifying the STC with the new definition of cluster score. This method is mainly focused on improving the effectiveness of document clustering. According to the preliminary experiment results, the new approach provides less number of base clusters. Then it ranks the clusters and more readable cluster labels are generated than that approach using the previous STC algorithms. This is the enhanced experimental evaluation of clustering algorithms on Web search engine results. The future work is extended to use the NLP for longer queries. It can also be combined with Word net for calculating the semantic similarity measures. A Fast algorithm is needed to process a large document collection and to decrease the response time.

5. REFERENCES

[1] L. Page and S. Brin, "The anatomy of a search engine", in. Proc. of the 7th International WWW Conference (WWW 98), Brisbane, Australia, April 14–18, 1998.

[2] Jansen, B. J, "The effect of query complexity on Web searching results", Information Research, Volume 6 No. 1, October, 2000.

[3] M. Liu, X. & Croft, W.B, "Statistical Language Modeling for Information Retrieval", In Cronin, B. (Ed.). Annual Review of Information Science & Technology. Vol 38, 2004.

[4] D. R. Cutting, D. R. Karger, J. O. Pedersen and J. W. Tukey, "Scatter/Gather: a cluster-based approach to browsing large document collections", In Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 318-29, 1992.

[5] Zamir O., Etzioni O, "Web Document Clustering: A Feasibility Demonstration", Proceedings of the 19th International ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'98), 46-54, 1998.

[6] Baeza-Yates, R., Ribeiro-Neto, B. Modern Information Retrieval. ACM Press. New York. pp. 25-30, 1999.

[7] Majumder P., Mitra M., Chaudhari B, "N-gram: A Language Independent Approach to IR and Natural Language Processing", Lecture Notes, 2002.

[8] Narita, M.& Ogawa, Y, "The use of phrases from query texts in information retrieval", SIGIR Forum, 34, 318-20 RIAO, College de France, pp. 665-681, 2000.

[9] Khaled M. Hammouda, Mohamed s. Kame, "Efficient Phrase-Based document Indexing for web document clustering", IEEE Transactions on Knowledge and Data Engineering, vol. 16, No. 10, Oct, 2004.

[10] Hua-Jun Zeng and et.at., "Learning to Cluster Web Search Results ", SIGIR'04 , Peking University, 2004.

[11] Hung, C. and D. Xiaotie, "A new suffix tree similarity measure for document clustering", In Proceedings of the 16th international conference on World Wide Web.ACM: Banff, Alberta, Canada, 2007.

[12] J.W.Yang, "A Chinese Web Page Clustering Algorithm Based on the Suffix Tree", Wuhan University Journal of National Sciences [M]. 9 (5):817-822, 2004

[13] Yahoo! Search BOSS (Build your Own Search Service) http://developer.yahoo.com/search/boss/

[14] A Vector Space Model For Automatic Indexing, G. Salton, A. Wong and C. S. Yang, Cornell University.

[15] SharpNLP - open source natural language processing tools, http://www.codeplex.com/sharpnlp

[16] M. F. Porter, "An algorithm for suffix stripping", Program, 14(3), pp.130-137, 1980.

[17] Salton, Gerald, and Christopher Buckley," Term-weighting approaches in automatic text retrieval", IP&M 24(5):513–523. 133, 520, 530, 1988.