Proceedings of the World Congress on Engineering and Computer Science 2012 Vol I
WCECS 2012, October 24-26, 2012, San Francisco, USA

Acyclic Word Graph for Web Clustering

Issam A.R. Moghrabi, *Member,* IAENG

Abstract- **The low precision of the Web search engines coupled with the ranked list presentation make it hard for users to find the information they are looking for. In this paper, the Web Document Clustering problem is studied. We propose a new data structure known as Compact Directed Acyclic Word Graph (CDAWG). The CDAWG is a space-efficient data structure used in handling and analyzing repetitions in a text. It is superior in terms of memory/space usage over Suffix Trees and ordinary DAWG especially on large data collection sets.**

Index Terms--Information retrieval, Acyclic Word Graph, clustering, suffix trees

I. INTRODUCTION

Document retrieval systems typically present search results in a ranked list, ordered by their estimated relevance to the query. The relevancy is estimated based on the similarity between the text of a document and the query. Such ranking schemes work well when users can formulate a well-defined query for their searches. However, users of Web search engines often formulate very short queries (70% are single word queries [1]) that often retrieve large numbers of documents. These problems are provoked when the users are unfamiliar with the topic they are querying about. Therefore, the vast majority of the retrieved documents are often of no interest to the user. Such searches are termed *low precision searches.*

Document clustering algorithms attempt to group similar documents together. Clustering the results obtained by Web search engines can provide a powerful browsing tool. The "*user-cluster hypothesis*" used in this research is that users have a mental model of the topics and subtopics of the documents present in the result set.

Similar documents will tend to belong to the same mental category in the users' model. Thus, the automatic detection of clusters of similar documents can help the user in browsing the result set.

The key requirements for document clustering of search engine results are:

1.Coherent Clusters: The clustering algorithm should group similar documents together.

2.Efficiently browsable: The user needs to determine at a glance whether the contents of a cluster are of interest. Therefore, the system has to provide concise and accurate cluster descriptions.

Manuscript received July 2, 2012; revised August 11, 2012.
I.A.R. Moghrabi is a professor of Computer Science at the M.I.S Department, Gulf University for Science and Technology, Kuwait (moughrabi.i@gust.edu.kw).

3.Speed: The clustering system should not introduce a substantial delay before displaying the results.

Response time of an information system can be improved by reducing the number of buckets accessed when retrieving a document set. One approach is to restructure the document base in such a way that similar documents are placed in close proximity in the document space.

In this work, we study CDAWG (Compact Directed Acyclic Word Graphs) Clustering which is well suited for large collection sets like web documents and is both memory and space efficient.

II. DOCUMENT CLUSTERING IN INFORMATION RETRIEVAL

Document clustering has initially been investigated in Information Retrieval mainly as a means of improving the performance of search engines by pre-clustering the entire corpus [2], [3], [4]; [5]. The *cluster hypothesis* [6] states that similar documents will tend to be relevant to the same queries, thus the automatic detection of clusters of similar documents can improve recall by effectively broadening a search request.

A second approach has been to group retrieval results using a pre-computed hierarchy of clusters of the entire corpus [7]. As the pre-computed clusters are not always "suitable" for the retrieval results, additional processing might be needed to improve the quality of the clusters [8]. This is done by first finding a set of clusters in the existing cluster hierarchy that represent a reasonable embedding of the retrieved documents. Large clusters are expanded and replaced by their children, resulting in a predefined constant number of clusters. A clustering algorithm is then applied to these predefined clusters (merging similar clusters together).

The CDAWG algorithm identifies phrases that are common in the document set, and use these phrases as the basis for creating clusters.

Phrases have long been used to supplement word-based indexing in IR systems. Syntactic phrases are generated using syntactic parsing to find words in a particular syntactic relationship. These techniques break down into two major categories: template-based and parser-based. Template-based techniques attempt to match adjacent words against a library of templates such as <JJ-NN NN> (adjective noun) and <NN PP NN> (noun preposition noun) [7]. Parser-based systems attempt to analyze entire sentences. Recently, the use of syntactic phrases has shown a consistent and significant improvement in retrieval performance (typically improving precision without hurting recall [9]).

The use of phrases or multiword features in document clustering is less common. Pairs of words that co-appear within a sliding window of five words (termed *lexical affinities*) were used as the attributes of the documents' vector representations (instead of single words) [10]. A standard HAC algorithm was then applied, producing better results than when using vector representations with single words as attributes.

III. COMPACT DIRECTED ACYCLIC WORD GRAPHS

The Directed Acyclic Word Graph (DAWG) is a space-efficient data structure to treat and analyze repetitions in a text, especially in DNA sequences and large data collection.

In this section we will utilize a new linear algorithm presented in [11] constructs Compact DAWG (CDAWG) as the data structure to hold the Web Document Suffixes.

Fast and space-economical methods for direct construction are important because the automaton serves as an index on the underlying word. Basically DAWGs provide an implementation of indexes on texts [12]. The index on a text *T* helps searching it for various patterns. The typical running time of a query is O(|w|) (for w = count of the query terms) on a fixed alphabet, and is O(|w|log|∑|) if the alphabet ∑ of the text is unbounded.

A. Definitions

In this section we will recall the basic notions on DAWG and CDAWG.

Definition1: The Suffix Automaton of a word x, denoted DAWG(x), is the minimal deterministic automaton that accepts S(x), the (finite) set of suffixes of x.

Definition 2: The size of the DAWG of a word is O(|x|) and the automaton can be computed in time O(|x|). The maximum number of states of the automaton is 2|x|-1, and the maximum number of edges is 3|x|-4.

Definition 3: Let u be a word of C, a class of factors of ≡S(x). If at least 2 letters a and b of ∑ exist such that ua and ub are factors of x, then C is called a strict class of factors of ≡S(x).

Below are the functions Endpos and length.
$\text{Endpos}_x(u) = \min\{|w| \mid w \text{ prefix of } x \text{ and } u \text{ suffix of } w\}$
$\text{Length}_x(p) = |u|$, with u representative of p.

Definition 4: Let p be a state of DAWG(x), different from the initial state, and let u be a word of the equivalence class. The suffix link of p, denoted by $s_x(p)$, is the state q which representative v is the longest suffix z of u such that $u \neq s_x(z)$.

B. Compact DAWG

Compaction of DAWG is based on the deletion of some states and their outgoing transitions [11]. This is possible by using multi-letter transitions and selecting strict classes of factors.

Definition 5: The CDAWG of a word x, denoted by CDAWG(x), is the compaction of DAWG(x) obtained by keeping only states that are either terminal states or strict classes of factors according to $\equiv_{S(x)}$, and by labeling transitions accordingly.

We achieve CDAWG by deleting every state having outdegree one exactly, except terminal states (Initial and Final states).

When a state p is deleted, the deletion of its outgoing edges is realized by concatenating their label to the labels of incoming edges. For example, let r and p be the states linked by a transition (r,b,p). The edges (r,b,p) and (p,a,q) are replaced by the edge (r,ba,q) if p is deleted.

Definition 6: If p is a state of CDAWG(x), the $s_x(p)$ is a state of CDAWG(x).

Theoretical average number of states calculated by [12], are 0.54n for CDAWG, 1.62n for DAWG and 1.62n for suffix trees, when n is the length of x. This gives respective sizes in bytes per character of the source: 45.68 and 32.70 for suffix trees, 33.62 and 27.80 for DAWG, and 22.40 and 22.78 for CDAWG.

We define a binary similarity measure between phrase clusters based on the overlap of their document sets. Given two phrase clusters m_i and m_j, with sizes $|m_i|$ and $|m_j|$ respectively, and $|m_i \cap m_j|$ representing the number of documents common to both phrase clusters, we define the similarity $\text{sim}(m_i,m_j)$ of m_i and m_j to be:
$\text{sim}(m_i,m_j) = 1$, if $|m_i \cap m_j| / |m_i| > \alpha$
and $|m_i \cap m_j| / |m_j| > \alpha$,
$\text{sim}(m_i,m_j) = 0$ otherwise,
where α is a constant between 0 and 1 (we typically use α = 0.6).

Next, we look at the *phrase cluster graph*, where nodes are phrase clusters, and two nodes are connected if and only if the two phrase clusters have a similarity of 1. A cluster is defined as being a connected component in the phrase cluster graph. We call these *merged clusters*. Each merged cluster contains the union of the documents of all its phrase clusters. Figure 1 illustrates the phrase cluster graphs of the six phrase clusters from Table 1, for α values of 0.6 and 0.7. Table 2 lists the final merged clusters identified from the based cluster graphs of Figure 1.

Table 1. Internal suffix tree nodes as phrase clusters

Node	***Phrase***	***Documents***
a	cat ate	1,3
b	ate	1,2,3
c	cheese	1,2
d	mouse	2,3
e	too	2,3
f	ate cheese	1,2

Table 2. Merged clusters in the phrase cluster graph

Figure	Cluster Number	Phrase clusters	Documents
(a)	1	a	1,3
	2	b	1,2,3
	3	d,e	2,3
	4	c,f	1,2
(b)	1	a,b,c,d,e,f,g	1,2,3
(c)	1	a	1,3
	2	d,e	2,3
	3	c,f	1,2

The phrase cluster graphs of the six phrase clusters from Table 1:

(a) for α = 0.7 there are four connected components is the graph, representing four merged clusters.

(b) for α = 0.6 there is a single connected component is the graph, representing one merged cluster.

(c) If the word *ate* had been in our stoplist, the phrase cluster *b* would have been discarded as it would have had a score of 0, and for α = 0.6 we would have had three connected components in the graph, representing three merged clusters.

In essence, in this step we are clustering the phrase clusters using the equivalent of a single-link clustering algorithm, where a predetermined minimal similarity between phrase clusters serves as the halting criterion. We do not encounter the undesired chaining effect of single-link clustering because in the realm of phrase clusters we typically find only small connected components.

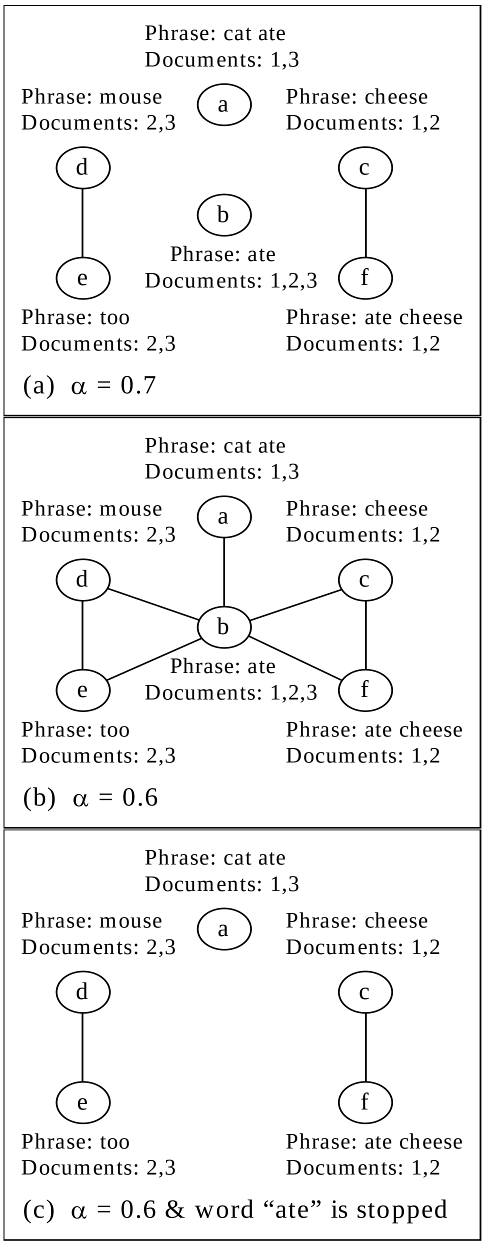

Figure 1. The phrase cluster graph

C. Constructing CDAWG

In this section, we give the direct construction of CDAWG based on [11]. The running time of the algorithm is linear in the size of the input word x on a fixed alphabet (see figure 2). The memory space is proportional to the size of the automaton.

Since the CDAWG of x is the minimization of its suffix tree, it is rather natural to base the direct construction on McCreight's algorithm [13].

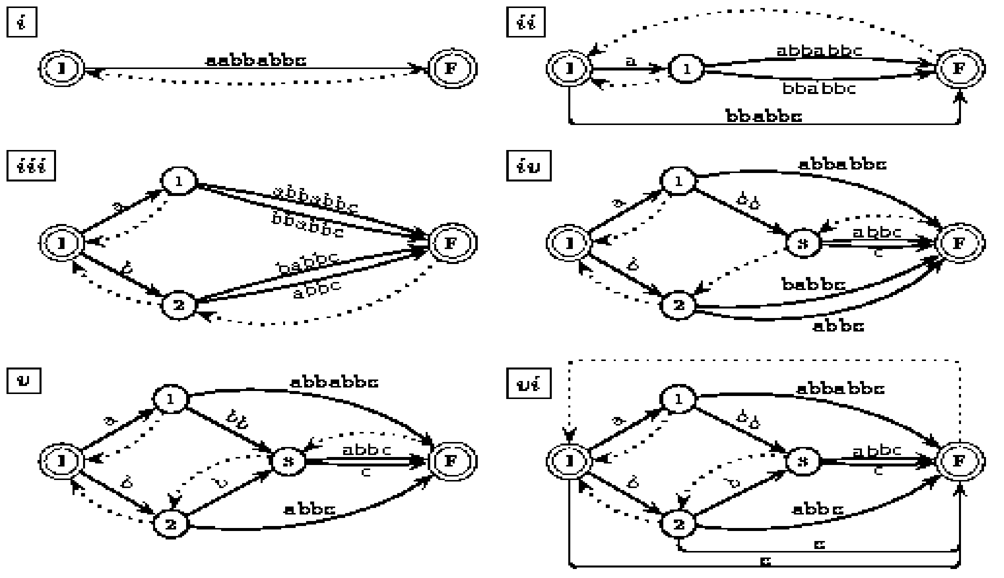

Figure 2. Six steps during the construction of CDAWG(aabbabbc)

We will restrict ourselves to presenting a rather general overview of the linear algorithm to construct CDAWG. For more information refer to [11].

```
Linear Algorithm
Input word x and threshold $\varepsilon$
1.  p ← I; i ←0;   //p is the current state
2.  While not end of x Do
3.    $(q,\gamma)$ ← Find(p);  //q state of the longest suffix
4.    If $(\gamma = \varepsilon)$ Then
5.      Insert (q,tail,F);
6.      $S_x(F)$ ← q;
7.      If $(q \neq I)$ Then p ← $s_x(q)$ Else p ← I;
8.    Else
9.      Create v locus of $head_I$ splitting $(q,\gamma)$;
10.     Insert the edge (v,tail,F);
11.     $s_x(F)$ ← v;
12.     find r = $s_x(v)$;
13.     p ← r;
14.   update i;
15. End While;
16. Mark terminal states
```

The algorithm time cost is O(|x|) and the corresponding space cost is O(|x| x card(∑)) using a transition matrix, or O(|x| x log card (∑)) and space cost of O(|x|) with adjacency lists.

The present structure provides an interesting space gain compared to the standard DAWG, and also when compared with suffix trees. From the theoretical point of view, the upper bounds are of |x| + 1 and 2|x| - 2 transitions. This saves |x| states and |x| transitions of the DAWG and at the same time leads to a faster use.

D. Implementation

The web document clustering process will proceed as mentioned by [14] mainly following the steps:

1. Document Cleaning
2. Identifying Base Clusters
3. Combining Base Clusters

The change will be in implementing the CDAWG data structure for storing all data pertaining to the documents and their associated phrases.

IV. CONCLUSIONS

We have considered the Compact Directed Acyclic Word Graph, which is an efficient compact data structure to solve the web document clustering problem. This structure provides an interesting space gain compared to Suffix Trees and standard DAWG. From the theoretical point of view, the upper bounds are |x|+1 states, and 2|x|-2 transitions. This saves |x| states and |x| transitions of the DAWG and at the same time leads to a faster use.

Web Document Clustering is a relatively new topic that is gaining interest worldwide due to the Internet revolution. Web document clustering can save time and effort and provide accuracy to the user who is searching the web. It provides a means for determining which cluster is of interest to the user.

Several enhancements can be included such as relevance feedback and the ability to use it online (at indexing time) taking into consideration the dynamic nature of the web.

REFERENCES

[1] H. Motro, Infoseek CEO, CNBC, May 7, 1998.

[2] N. Jardine and C. J. Van Rijsbergen,." The use of hierarchical clustering in information retrieval", *Information Storage and Retrieval*, 7:217-240, 1971.

[3] G. Salton, "Cluster search strategies and the optimization of retrieval effectiveness", In Salton, G. (ed), *The SMART Retrieval System*, Prentice-Hall, Englewood Cliffs, N.J., 223-242, 1971.

[4] W. B. Croft, "Organizing and searching large files of documents". Ph.D. Thesis, University of Cambridge, 1978.

[5] A. Griffiths, H. C. Luckhurst, and P., Willet, "Using inter-document similarity information in document retrieval systems". *Journal of the American Society for Information Science*, 37:3-11, 1986.

[6] C. J. Van Rijsbergen, *Information Retrieval*, Butterworths, London, 1979.

[7] D. R Cutting,., Karger, D. Pedersen, J. O. Tukey and J. W. 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* (SIGIR'92), 318-329, 1992.

[8] C. Silverstein, and J. O Pedersen,. "Almost-constant time clustering of arbitrary corpus subsets". In *Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR'97), 60-66, 1997.

[9] Zhai, C., Tong, X., Milic-Frayling N. and Evans, D. A. "Evaluation of syntactic phrase indexing - CLARIT NLP track report". In: D. K. Harman (ed.), The Fifth Text Retrieval Conference (TREC-5). NIST Special Publication, 1997.

[10] Maarek, Y. S. and Wecker, A. J. "The Librarian's Assistant: Automatically organizing on-line books into dynamic bookshelves". In *Proceedings of the 1994 RIAO Conference* (RIAO'94), 1994.

[11] M. Crochemore and R. Verin, "On Compact Directed Acyclic Word Graphs", *Institut Gaspard Monge, Universite de Marne-La-Vallee,* http://www-igm.univ-mlv.fr.

[12] A. Blumer, J. Blumer, D. Haussler, and R. McConnell. "Complete inverted files for efficient text retrieval and analysis". *Journal of the Association for Computing Machinery,* 34(3):578-595

[13] E. M. McCreight, "A space-economical suffix tree construction algorithm". *Journal of the ACM,* 23:262-272, 1976.

[14] O. Zamir, , D. Etzioni, F. Madani and R.M. Karp, R. M. "Fast and intuitive clustering of Web documents". In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (KDD'97), 287-290, 1997.