Applying Semantic Suffix Net to Suffix Tree Clustering

Jongkol Janruang, Sumanta Guha
Computer Science & Information Management Program
Asian Institute of Technology
P.O. Box 4, Klong Luang
Pathumthani 12120
Thailand
Email: [jongkol.janruang, guha]@ait.asia

Abstract—In this paper we consider the problem of clustering snippets returned from search engines. We propose a technique to invoke semantic similarity in the clustering process. Our technique improves on the well-known STC method, which is a highly efficient heuristic for clustering web search results. However, a weakness of STC is that it cannot cluster semantic similar documents. To solve this problem, we propose a new data structure to represent suffixes of a single string, called a Semantic Suffix Net (SSN). A generalized semantic suffix net is created to represent suffixes of a set of strings by using a new operator to partially combine nets. A key feature of this new operator is to find a joint point by using semantic similarity and string matching; net pairs combination then begins at that joint point. This logic causes the number of nodes and branches of a generalized semantic suffix net to decrease. The operator then uses the line of suffix links as a boundary to separate the net. A generalized semantic suffix net is then incorporated into the STC algorithm so that it can cluster semantically similar snippets. Experimental results show that the proposed algorithm improves upon conventional STC.

Index Terms—semantic suffix net, semantic web search results clustering, data mining, text mining

I. INTRODUCTION

A suffix tree is a data structure that can provide efficient solutions, such as string searching and extracting the pattern of strings, for a myriad of string processing problems [1]. Suffix trees can also be used in applications such as document clustering [2], virus classification and virus identification [3]. In 1998, Oren Eli Zamir adapted suffix trees for use with a web document clustering algorithm called, Suffix Tree Clustering (STC) [2].

STC uses a suffix tree to represent document sets. The suffix tree is constructed by feeding a string of words to identify document sets that share common phrases. In detail, STC has three main steps. Step one is Document Cleaning; STC uses a light stemming algorithm to transform the string of text representing each document (deleting word prefixes and suffixes and reducing plural to singular) and skips the non-word tokens such as number, html tags and most punctuation. Step two is Identifying Base Clusters; STC feeds a string of words from all the sentences of all the documents in a document set to construct a generalized suffix tree. Step three is Combining Base Clusters; STC uses the base cluster graph

of a single-link clustering algorithm to avoid the proliferation of nearly identical clusters. As a result, STC is a highly efficient heuristic for document clustering. However, STC cannot deal with the semantic duplication of word pairs. This is because STC uses a suffix tree structure that used only string matching to construct tree for representing document sets. To overcome this difficulty, this paper proposes a semantic suffix net and a generalized semantic suffix net that works on strings of words by using both string matching and semantic similarity to construct a net structure.

WordNet [4], a large lexical database of English currently being developed, groups synonym words into a set called *synset*. *Synsets* returned by WordNet can be used in a number of semantic clustering algorithms such as the semantic Lingo algorithm [5] and the CFWMS algorithm [6]. This paper uses the *synset* to determine the semantic similarity of the word pairs and uses this information to construct a semantic suffix net and to generalize a generalized semantic suffix net.

This paper is organized as follows. Related work is covered in section II. Semantic Suffix Net with Suffix Tree Clustering is proposed in section III. Experiments are presented in section IV and the conclusion in section V.

II. RELATED WORK

The STC algorithm is a highly efficient heuristic for document clustering that can be used to solve the problem of long list results produced by search engines. STC uses a suffix tree structure to find a set of documents that share common phrases and uses this information to propose clusters. As a result, STC is fast at automatic clustering and labeling but it cannot cluster semantically similar snippets. For example, the two snippets s_1 =*doctor talks with nurse* and s_2 =*physician speaks to nanny* have similar meanings but STC cannot group them into the same cluster. In addition, text document and web document clustering related to web search results clustering is trending towards semantic clustering. Semantic clustering involves the actual meaning of words being used as a method of clustering and is used by algorithms such as the semantic Lingo algorithm [5], the Word Sense Induction algorithm [7] and the CSUGAR algorithm [8].

In recent years, several approaches have been proposed to solve the problems of the STC algorithm. First, the SHOC algorithm uses suffix arrays to extract frequent phrases and then a singular value decomposition technique is used to discover the cluster content [9]. This approach was taken because the original STC ignored the semantics of the word to propose clusters. However, SHOC does not support incremental processing, an important characteristic of web search results clustering. Second, the NSTC [10] algorithm was developed by combining the vector space model and suffix tree model to calculate the similarity of document pairs for solving the lack an efficient similarity measure of the original STC that causes a large-size cluster with poor quality to return. The NSTC algorithm involves a complicated implementation process and like the SHOC algorithm, does not support incremental processing. Third, the STC with n -gram technique [11] was introduced to manage the huge tree returned by the original STC but the difficulty with this approach is that the n -gram technique generates interrupted cluster labels if the common phrase size is greater than the defined n -gram size. Fourth, the STC with a partial phrase join operator [12] was presented to solve the problem of the interrupted cluster labels that are generated when using the STC with n -gram technique. Fifth, the STC with x -gram [13] was created based on the on-line construction of suffix trees to decrease the memory space used by the original suffix trees. Sixth, the STC with WordNet [14] was introduced to create a hierarchy of clusters using WordNet to compare the *synset* of each final cluster pair returned by the original STC. This was due to the original STC being unable to create a hierarchy of clusters for direct use. However, these several approaches do not modify the original STC to do semantic search results clustering.

To deal with these ignored semantics, this paper proposes a new data structure called a semantic suffix net (SSN) and a generalized semantic suffix net (GSSN) that is constructed by combining semantic suffix net pairs. These structures are a modification work of semantic suffix tree structure, introduced by Junruang and Guha [15] in order to address the problem of a lack the logic to explain the disappearance of semantically similar nodes. The GSSN is applied to replace the suffix tree in the original STC, called STC+GSSN. Based upon evaluations, GSSN can be used instead of the suffix tree structure in the original STC because STC+GSSN can cluster semantic similar snippets and the cluster quality that is returned by STC+GSSN is similar to the original STC.

III. SEMANTIC SUFFIX NET WITH SUFFIX TREE CLUSTERING

When semantic similar snippets are returned by a current search engine, identification of clusters should consider not only string matching but semantic similarity of words also. This is because the group of snippets can be changed after using semantic similarity to consider the cluster of snippets. A new data structure created using both string matching and semantic similarity is discussed in section III-A. Section III-B describes how to generate a generalized semantic suffix net

and section III-C describes how a semantic suffix net is used instead of a suffix tree in the suffix tree clustering algorithm.

A. Semantic Suffix Net

An SSN is a new data structure that uses both string matching and semantic similarity to create a net form. SSN uses semantic similarity measure of a semantic suffix tree structure [15] to construct a SSN and a GSSN structure.

Let S be a string with m -word strings where the elements of S are words $S = \{w_1, w_2, \dots, w_m\}$. An β denotes the pattern of the suffix created depending on word sequences, which are extremely important in determining the meaning of natural languages, and ϵ denotes an *empty string*. Nodes that represent each word are denoted as v and $s[v]$ as a suffix node of v . Let *suffix link* be a connected link from node v to $s[v]$ that connects between β_i and β_j . The term *semantic link* is a link from $s[v_i]$ to $c[v_{i-1}]$ when $c[v_{i-1}]$ is a child node of v_{i-1} . A *semantic link* is created if and only if $s[v_i]$ and $c[v_{i-1}]$ are matched by semantic similarity.

Definition: A SSN for an m -word string S is a rooted directed net with leaves $\leq m$ attached to the starting position in that string and has a suffix link from v to $s[v]$ of $S[2..n]$ between β_i to β_j . This is only true however, if the word pairs do not match by string or by semantic similarity. The start of each β_{i+1} is created by a second node of β_i and then directed by a root node. The key feature of a SSN is that it simultaneously constructs the semantic suffix net through an on-depth and on-breadth pass by using string matching and semantic similarity.

Algorithm (III-A) shows that it is possible to construct SSN on-depth and on-breadth in a left-to-right scan over S as follows.

The start state $g(v_{i-1}, v_i)$ is defined for translating each transition state to create the SSN on-breadth pass suffix link $sl(v_i, s[v_i])$ and directed link $dl(s[v_{i-1}], s[v_i])$. But only if $s[v_i]$ does not match by string with children node of $s[v_{i-1}]$. Following this, a semantic link $sel(s[v_i], c[v_{i-1}])$ is only created if $s[v_i]$ is matched with children node of v_{i-1} by semantic as shown in line 19. If however, v_{i-1} does not have a suffix node then a root node is a suffix node of v_{i-1} when ϵ in line 17 means *empty string*. For the last variable in line 24, l denotes the starting position of the suffix in that string.

[Algorithm III-A The construction of a semantic suffix net for a single string S]

1. Let $S \leftarrow \{w_1, w_2, \dots, w_m\}$
2. Assign $i \leftarrow 1$
3. Create $v_i \leftarrow w_i$
4. Create $dl(root, v_i)$
5. For $i \leftarrow i + 1$ to m do
6. Create $v_i \leftarrow w_i$
7. If $v_i \approx c[v_{i-1}]$ Then Create $dl(v_{i-1}, v_i)$
8. Else $c[v_{i-1}] \leftarrow v_i \oplus c[v_{i-1}]$
9. Assign start state $g(v_{i-1}, v_i)$
10. While $v_{i-1} \in g(v_{i-1}, v_i) \neq root$ do
11. Create $s[v_i] \leftarrow v_i$

```

12.   If  $s[v_i]! \approx c[s[v_{i-1}]]$  Then Create  $sl(v_i, s[v_i])$ 
13.   Else {
14.        $c[s[v_{i-1}]] \leftarrow s[v_i] \oplus c[s[v_{i-1}]]$ 
15.       Create  $sl(v_i, c[s[v_{i-1}]])$ 
16.   }
17.   If  $s[v_{i-1}] \neq \epsilon$  Then  $v_{i-1} \leftarrow s[v_{i-1}]$ 
18.   Else  $v_{i-1} \leftarrow root$ 
19.   If  $s[v_i] \approx c[v_{i-1}]$  Then Create  $sel(s[v_i], c[v_{i-1}])$ 
20.   If  $s[v_i]! \approx c[v_{i-1}]$  Then Create  $dl(v_{i-1}, s[v_i])$ 
21.   Else  $c[v_{i-1}] \leftarrow s[v_i] \oplus c[v_{i-1}]$ 
22.   Change start state  $g(v_{i-1}, v_i) \leftarrow g(v_{i-1}, s[v_i])$ 
23. } //end of while
24. If  $i \geq m$  Then Add  $l$  into  $m$  leaves
25. } //end of for

```

Figure (1) shows the different phases of constructing $SSN(S)$ for $S = \text{doctor talks to nurse and physician}$ and Figure (2) is shows the whole semantic suffix net which is created by using Algorithm (III-A).

Fig. 1. This diagram shows how to simultaneously construct an SSN through an on-depth and on-breadth pass by using a suffix link and a directed link. Starting with $g(v_{i-1}, v_i)$ to build the SSN on-breadth, it is then converted at the next stage until v_{i-1} is a root node.

B. A Generalized Semantic Suffix Net

The SSN can be used to present suffixes of a single string and can be extended to present suffixes of a set of strings in a net form, called a generalized semantic suffix net (GSSN).

Definition: A GSSN presents a set of strings with m -word string S_k and is derived from the combination of a semantic suffix of net pairs using a partial combine nets operator.

In the Algorithm (III-B), the construction of $SSN(S_k)$ on line 2 and 4 is similar to Algorithm (III-A) but this construction must add a two number tuple (k, l) where k ranges from 1 to n and l ranges from 1 to m_k . The construction of a GSSN exists in line-5 which uses a partial combine nets operator to combine the SSN pairs.

[Algorithm III-B The construction of a GSSN structure for a set of strings]

1. Let $S \leftarrow \{s_1, s_2, \dots, s_n\}$
2. Construct $SSN(s_1) \leftarrow s_1$
3. For $i \leftarrow 2$ to n do

Fig. 2. This diagram shows a complete semantic suffix net. The box of a leaf node, a starting position of the suffix in that string, is added to leaves.

4. Construct $SSN(s_{i+1}) \leftarrow s_{i+1}$
5. $SSN(s_i) \leftarrow SSN(s_i) \oplus SSN(s_{i+1})$

To combine the SSN pairs, let β_p denotes the suffix $\beta_1, \beta_2, \dots, \beta_p$ of $SSN(S_i)$ when $p = 1 \dots b$ and b is a number of suffixes of $SSN(S_i)$. Let β_q be the suffix $\beta_1, \beta_2, \dots, \beta_q$ of $SSN(S_{i+1})$ for $q = 1 \dots m$ and m is a number of suffixes of $SSN(S_{i+1})$. Let n be the height of a $\beta_1(SSN(S_{i+1}))$. The first node of remainder part of β_q is defined as a center point cp when it is a member of $SSN(S_i)$ after $\beta_q(SSN(S_{i+1}))$ is combined with $\beta_p(SSN(S_i))$. A remainder part of $SSN(S_{i+1})$ is a lower boundary that is separated by the line of *suffix link* of a node that is matched with a joint point jp .

A partial combine nets operator uses semantic similarity and string matching to find a longest common concept LCC that is shared by $\beta_p(SSN(S_i))$ and $\beta_1(SSN(S_{i+1}))$. Therefore, LCC is a number of the word in a longest common concept that is used to determine a joint point jp on $SSN(S_i)$. For example, $\beta_p(SSN(S_i)) = \{\text{doctor, care, baby}\}$ and $\beta_1(SSN(S_{i+1})) = \{\text{doctor, care, patient}\}$ as a result, $LCC = \{\text{doctor, care}\}$ and jp is a node number 2 on $\beta_p(SSN(S_i))$. Subsequently, jp is used to find a new joint point by comparing a suffix node between jp and cp . If cp and jp do not have a suffix node then $root$ is a new joint point else a suffix node of jp or cp is a new joint point.

A GSSN is derived from a combination of the SSN pairs by adhering to the following three rules:

Rule1: $LCC = n$ thus a node number n of $SSN(S_i)$ is a jp . To update the $SSN(S_i)$, the start of updating the $SSN(S_i)$ is added tuple (k, l) of $\beta_1(SSN(S_{i+1}))$ into jp and then all suffix nodes of a joint point $s[jp]$ are updated alike jp by traveling pass *suffix link*.

Rule2: $1 \leq LCC < n$ thus a node number LCC is a jp . To update the $SSN(S_i)$, the remainder part of $SSN(S_{i+1})$ is

For example, using four strings, $S_1 = \textit{doctor cares baby}$, $S_2 = \textit{physician likes child}$, $S_3 = \textit{physician cares patient}$, and $S_4 = \textit{emergency doctor guide}$, to construct a GSSN using Algorithm (III-B). Figures 3, 4, and 5, show the working steps of the three rules, respectively.

Fig. 3. $SSN(S_i) \leftarrow jp \oplus (k, l)(\beta_1(SSN(S_{i+1})))$ is the beginning of updating $SSN(S_i)$ and then $SSN(S_i) \leftarrow s[jp] \oplus (k, l)(\beta_j(SSN(S_{i+1})))$ when $j = 2..m$ is a next step to update $SSN(S_i)$. There is the logic of *Rule1*: $LCC = n$.

Combining $SSN(S_i)$ and $SSN(S_{i+1})$ make it easier to combine the net pairs because these structures are consistent with the *suffix link*. That is, after the algorithm has found a joint point, the algorithm can use the suffix link to get only the remainder net of $SSN(S_{i+1})$ to update the $SSN(S_i)$.

C. Applying a generalized semantic suffix net to suffix tree clustering

To deal with the problem of the original STC algorithm, a highly efficient heuristic for web search result clustering, a GSSN is used instead of a suffix tree to represent the snippet set $S = \{s_1, s_2, \dots, s_n\}$ in the original STC [2]. This algorithm is called STC+GSSN and is created to solve the problem of STC's inability to cluster semantically similar snippets. The STC+GSSN has three logical steps.

Step1: document cleaning, STC+GSSN returns a word to root of the word by using a light stemming algorithm and skips stop-word and non-word tokens such as *a, an, the, numbers, html tags and most punctuation*.

Step2: identify base clusters using a GSSN. A GSSN for all suffixes of each snippet in snippet sets $S = \{s_1, s_2, \dots, s_n\}$ is constructed. Each node containing at least two different documents is selected to be a base cluster. A base cluster is composed of a document number and a position of string, designated by the box, and labeled by the phrase of the node. The score of each base cluster is assigned using Equation (1).

Fig. 4. $SSN(S_i) \leftarrow R(\beta_q(SSN(S_{i+1}))) \cup R(\beta_p(SSN(S_i)))$ when $q = 1 \dots m$. This is the logic of the *Rule2*: $1 \leq LCC < n$.

$$S(B) = |B| \cdot \int (|P|) \quad (1)$$

When $|B|$ is the number of documents in a group B , and $|P|$ is the number of words in phrase P when the function $\int$ penalizes single word phrases and becomes a constant for longer phrases. All base clusters are sorted by their scores and then merged in next step.

Step3: identify the final cluster by combining the base clusters; a similarity measure of two base clusters is calculated using the base cluster graph concept to define the similarity of B_m and B_n to be 1 iff: $|B_m \cap B_n|/|B_m| \geq 0.5$ and $|B_m \cap B_n|/|B_n| \geq 0.5$; otherwise, their similarity is defined to be 0. An example of the base cluster graph concept as

Fig. 5. $SSN(S_i) \leftarrow \beta_1(SSN(S_{i+1})) \oplus SSN(S_i)$ is a start step to update $SSN(S_i)$ and then $SSN(S_i) \leftarrow R(\beta_q(SSN(S_{i+1}))) \cup R(\beta_p(SSN(S_i)))$ when $q = 2 \dots m$ that is a next step to update $SSN(S_i)$. This is the logic of the Rule3: $LCC = 0$

shown in Figure 6.

The three logical steps of STC+GSSN are similar to the original STC but STC+GSSN uses a GSSN instead of a suffix tree structure. For example, clustering the following four snippets, $S_1 = \text{doctor cares baby}$, $S_2 = \text{physician likes child}$, $S_3 = \text{physician cares patient}$, and $S_4 = \text{emergency doctor guide}$ using STC+GSSN returns six base clusters. These clusters are: $\text{doctor} = \{1,2,3,4\}$, $\text{care} = \{1,2,3\}$, $\text{baby} = \{1,2\}$, $\text{doctor care} = \{1,2,3\}$, $\text{care baby} = \{1,2\}$, and $\text{doctor care baby} = \{1,2\}$ by checking the member of each node, if a node has a member more than one then it is a group document.

From Figure 6, in this example there is one final cluster because there is one connected component. As a result, a final cluster is $\text{doctor care} = \{1,2,3,4\}$. For the final cluster, all base clusters are merged because their members are related by more than 0.5, which is a minimum threshold. These results show that STC+GSSN can cluster semantic similar snippets although their strings do not match. The strings $S_1 = \text{doctor cares baby}$ and $S_2 = \text{physician likes child}$ do not match by string but they match by meaning. STC+GSSN can therefore solve the problem of STC's inability to cluster semantic similar snippets.

IV. EXPERIMENTAL RESULTS

Table 1 compares the clusters that are generated using conventional STC with the clusters that are generated using STC+GSSN. It is shown that STC+GSSN can cluster the

Fig. 6. this diagram shows a similarity measure of base clusters is calculated using the base cluster graph concept of original STC after a GSSN returns base clusters and their score are returned in second step.

snippets that share a semantic similarity and it can return the cluster label in a more readable form than conventional STC. In addition, we tried to change the minimum threshold to 0.5 and 0.75 for checking the different results. As a result, size of cluster and number of cluster is depended on identifying minimum threshold value. This is because the size of clusters of minimum threshold at 0.5 is bigger than 0.75 but the number of cluster of minimum threshold at 0.5 is less than 0.75. The difficulty is that the suitable minimum threshold value is difficult to identify since low value of minimum threshold returns some large-sized clusters with poor quality that causes a time consuming task to find the relevant ones. On the other hand, small-sized clusters with rich quality is returned if minimum threshold is high but large number clusters causes a time consuming task also.

The dataset for testing the search results clustering engine is created from a small testing dataset using 26,890 search results from 10 queried words on Dmoz.com [16]. Table 2 compares the performance between conventional STC and STC+GSSN.

TABLE I
COMPARING THE RESULTS BETWEEN CONVENTIONAL STC AND
STC+GSSN IN THE DIFFERENCE MINIMUM THRESHOLD

Document title	Min.	Comparison of clusters	
		STC+GSSN	STC
1.doctor cares baby	0.5	doctor care 1,2,3,4	doctor 1,2,3,4
2.physician like child	0.75	doctor care baby	doctor 1,4
3.physician cares patient		1,2	care 1,3
4.emergency doctor guide		doctor care 1,2,3,4	physician 2,3

TABLE II
A COMPARISON OF FIVE MEASURES OF EFFICIENCY AND
EFFECTIVENESS BETWEEN CONVENTIONAL STC AND
STC+GSSN

Algorithm	Precision	Coverage	Num. of Clusters	Num. of nodes	Num. of branches
STC	0.68	1	5,311	358,959	8,065
STC+GSSN	0.67	1	4,686	353,637	6,717

The precision measures between STC and STC+GSSN decreases because identifying final cluster in conventional STC uses similarity measure equation that consider only overlap of documents in the cluster pairs. As a result, some large-sized clusters with poor quality are returned. However, the coverage measure is unique and the number of clusters, nodes, and branches measured in STC+GSSN are decreased. This points to greater efficiency over the original STC.

V. CONCLUSION

This paper proposes a new data structure, called SSN to present the suffixes of a single string and extended SSN to present the suffixes of a set of strings in the net form, called a GSSN. A GSSN can be used instead of a suffix tree in a suffix tree clustering algorithm. This algorithm is called STC+GSSN and it can cluster snippets that share a semantic similarity. Based on evaluations, STC+GSSN can return cluster labels in a readable form and the cluster quality is similar to the original STC. The number of clusters, nodes, and branches are reduced meaning that efficiency is greater than that of the original STC. Therefore, a GSSN can be used to improve the performance of approaches that use a suffix tree to represent data for web search results clustering algorithms and text document clustering. This is because a GSSN structure can modify the features of data presentation to become a semantic clustering algorithm. However, STC+GSSN returned some large-sized clusters with poor quality because these large-sized clusters with poor quality are generated by identifying the final cluster of the original STC which lacks an efficient similarity measure.

Future plans are to extend the generalized semantic suffix net to create a new incremental clustering algorithm for improvement of the STC+GSSN that returned the large-sized clusters with poor quality.

REFERENCES

- [1] Gusfield, D.: Algorithms on String, Trees, and Sequences computer science and computational biology. Cambridge University Press (1997)
- [2] Zamir, O., Etzioni, O.: Web document clustering: a feasibility demonstration. In: SIGIR 1998, pp. 46-54 (1998)

- [3] Sheng-Lung, P. Jyun-Jhao, C.: Classifying and Identifying HPV Based on Maximal Common Subsequences. In: Journal of Computers Vol.18, No.3, October 2007 (2007)
- [4] a large lexical database of English, <http://wordnet.princeton.edu/>
- [5] Ahmed, M.S., Amar, M.K.: Semantic Web Search Results Clustering Using Lingo And Wordnet. In: IJRRCS: Kohat University of Science and Technology (KUST), Vol. 1, No 2, pp. 71-76. , Pakistan (2010)
- [6] Li, Yanjun and Chung, Soon M. and Holt, John D.: Text document clustering based on frequent word meaning sequences. In: Journal Data Knowledge Engineering. volume 64, pp.381-404 (2008).
- [7] Navigli, R., Crisafulli, G.: Inducing Word Senses to Improve Web Search Result Clustering. In: Proc. EMNLP, MIT Stata Center, Massachusetts, USA, pp. 116-126 (2010).
- [8] Illhoi, Y., Xiaohua, H., Il-Yeol, S.: A coherent graph-based semantic clustering and summarization approach for biomedical literature and a new summarization evaluation method. In: BMC Bioinformatics 2007, 8(Suppl 9):S4 (2007)
- [9] Dell, Z., Yisheng, D.: Semantic, Hierarchical, Online Clustering of Web Search Results. In: APWeb 2004, pp.69-78 (2004)
- [10] Chim, H., Deng, X.: A new suffix tree similarity measure for document clustering. In: WWW 2007, pp. 121-130., ACM, New York, NY, USA (2007)
- [11] Zeng, H., He, Q., Chen, Z., Ma, W., Ma, J.: Learning to cluster web search results. In: SIGIR '04, pp. 210-217., ACM, New York, NY, USA (2004)
- [12] Janruang, J., Kreesuradej, W.: A New Web Search Result Clustering based on True Common Phrase Label Discovery. In: CIMCA '06, pp. 242, IEEE Computer Society, Washington, DC, USA (2006)
- [13] Wang, J., Mo, Y., Huang, B., Wen, J., He, L.: Web Search Results Clustering Based on a Novel Suffix Tree Structure. In: ATC '08, pp. 540-554, Springer-Verlag, Berlin, Heidelberg (2008)
- [14] Guodong, h., Wanli, Z., Fengling, H., Ying, W.: Semantic-Based Hierarchicalize the Result of Suffix Tree Clustering. In: Second International Symposium on Knowledge Acquisition and Modeling, pp.221-224 (2009)
- [15] Janruang, J., Guha, S.: Semantic Suffix Tree Clustering. In: DEIT 2011, IEEE, Bali, Indonesia (2011).
- [16] Open Directory Project, <http://www.dmoz.com>