WordNet-Based Suffix Tree Clustering Algorithm

Qiuyue Dang[1,a], Jiwei Zhang[1,b], Yueming Lu[2,c] and Kuo Zhang[3,d]

[1]School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China

[2]Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing, China

[3]Beijing Sogou Technology Development Co., Ltd, Beijing, China

[a]qiuyue_dang@163.com, [b]p40.fate@gmail.com , [c]ymlu@bupt.edu.cn, [d]zhangkuo@sogou-inc.com

Keywords: search results clustering, suffix tree, STC, WordNet synsets.

Abstract. High space cost and ignoring synonyms in STC (Suffix Tree Clustering algorithm) are challenges for search results clustering. Aiming at these challenges, this paper proposes a WordNet-based suffix tree clustering algorithm (WNSTC). WNSTC can construct a suffix tree containing WordNet synsets. When constructing the suffix tree, WNSTC looks every feature word up in WordNet database. If the feature word is included in WordNet, its synsets will be added into corresponding node. The node in the suffix tree may be a set of words (strings) with similar meaning instead of a single word (string). Experiments executed on data sets show that WNSTC has better clustering quality and smaller suffix tree size than original STC algorithm.

Introduction

Search engines are considered as the most common tool to retrieve information from the Internet. But the search results returned by search engines usually come with huge quantity and a long list. What's more, the search results will be very diverse when users' search term is not correct or ambiguous. It's time-consuming and arduous to find the most relevant search result [1]. Search results clustering is an efficient method to make the search results easier to scan. It aims to cluster search results into meaningful groups and provide a navigator to easily access the satisfying results for users [2]. Search results clustering works on snippets (a summary of the search results), which is different from document clustering (working on long text).STC (Suffix Tree Clustering) algorithm is a classic search results clustering algorithm, which was proposed by Zamir and Etzionit in 1998. It is an incremental and liner time (in the document collection size) algorithm, which creates clusters based on phases shared between documents [3]. STC doesn't use VSM (Vector Space Model) to express text as bag-of-words, which is different from other clustering algorithms, such as K-means [4]. Zamir and Etzionit showed that STC is faster and more flexible than other standard snippets clustering methods.

However, STC is still not perfect. It ignores the semantic information in snippets and in high-cost of space when the number of snippets is huge. There are two classic improved algorithms based on STC, SHOC [5] and Lingo [6]. SHOC uses singular value decomposition (SVD) to discover the semantic information in term-document matrix generated by VSM. Lingo uses common phase discovery and LSI (Latent Semantic Indexing) [7] to group snippets into meaningful clusters. Unfortunately, because their processes are still based on VSM, the semantic relationship between the words is not recognized explicitly. What's more, they are high-dimensional when applied to large numbers of snippets, which goes against the high space cost of STC.

In this paper, a WordNet-based suffix tree clustering algorithm is proposed, which is called WNSTC. WordNet ("WN" for short) - a large lexical database of English, is used in WNSTC to analyze the synonyms information in snippets explicitly [8]. In the feature extraction stage of WNSTC, only the nouns and verbs in snippets are extracted because they have great thematic significance in linguistics. This method is good for feature dimension reduction. When constructing the suffix tree, every feature word is retrieved from WordNet to get its synsets and we record them in the corresponding node. In other words, a few of conventional suffix tree nodes with different forms, which have the same meaning, will be one node in WNSTC. The effectiveness of WNSTC is proved by experiments in section 3.

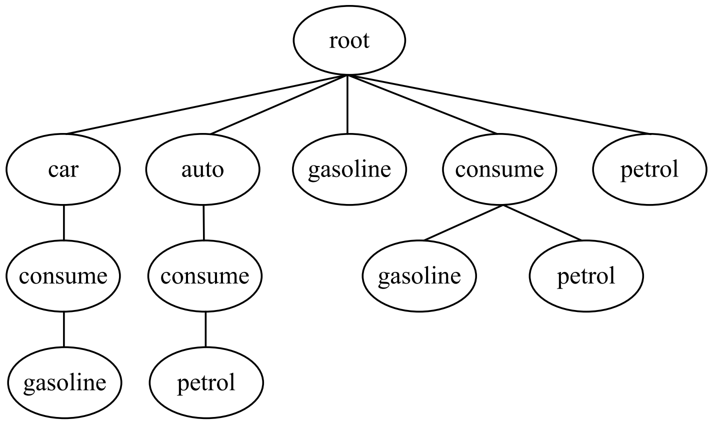


Fig. 1 The conventional suffix tree constructed by STC

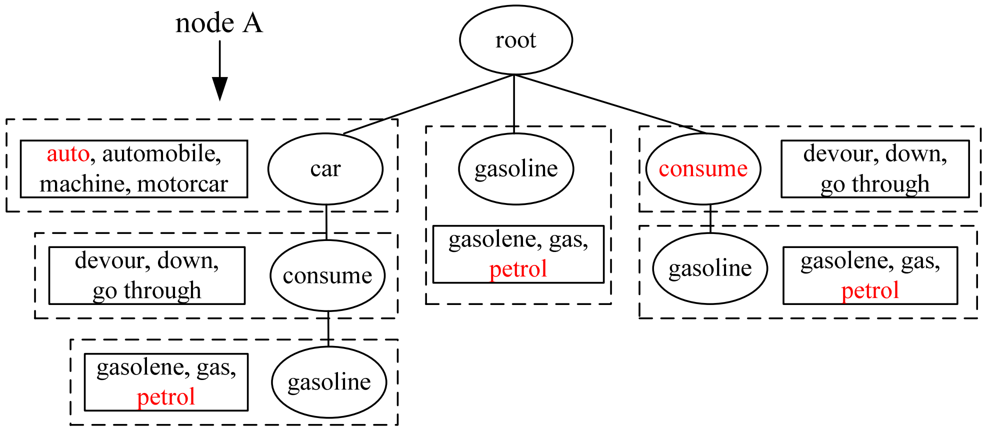


Fig. 2 The new suffix tree constructed by WNSTC

WordNet-Based Suffix Tree Clustering Algorithm

The Suffix Tree of WNSTC. WNSTC focuses on improving the clustering quality by integrating WordNet's synsets on the semantic level. For this reason, the suffix tree of WNSTC is designed to highlight the similar meaning expressed by synonyms for the string. Starting from this idea, we add the feature words' synsets to the suffix tree, which are retrieved from WordNet database. Here is an example. Supposing we have a feature word – car, its synsets in WordNet are listed below:

a) car, auto, automobile, machine, motorcar
b) car, railcar, railway car, railroad car
c) cable car, car
d) car, gondola

e) car, elevator car

The first synset is the most common meaning of “car”. In order to reduce the complexity of nodes, we just choose the first synset to add to the node standing for “car”. Now, the node in WNSTC’s suffix tree will not indicate a word or a string with the same characters, but a similar meaning expressed by a set of synonymous words or strings, like {car, auto, automobile, machine, motorcar} . But, we still use “car” as the main word for this node.

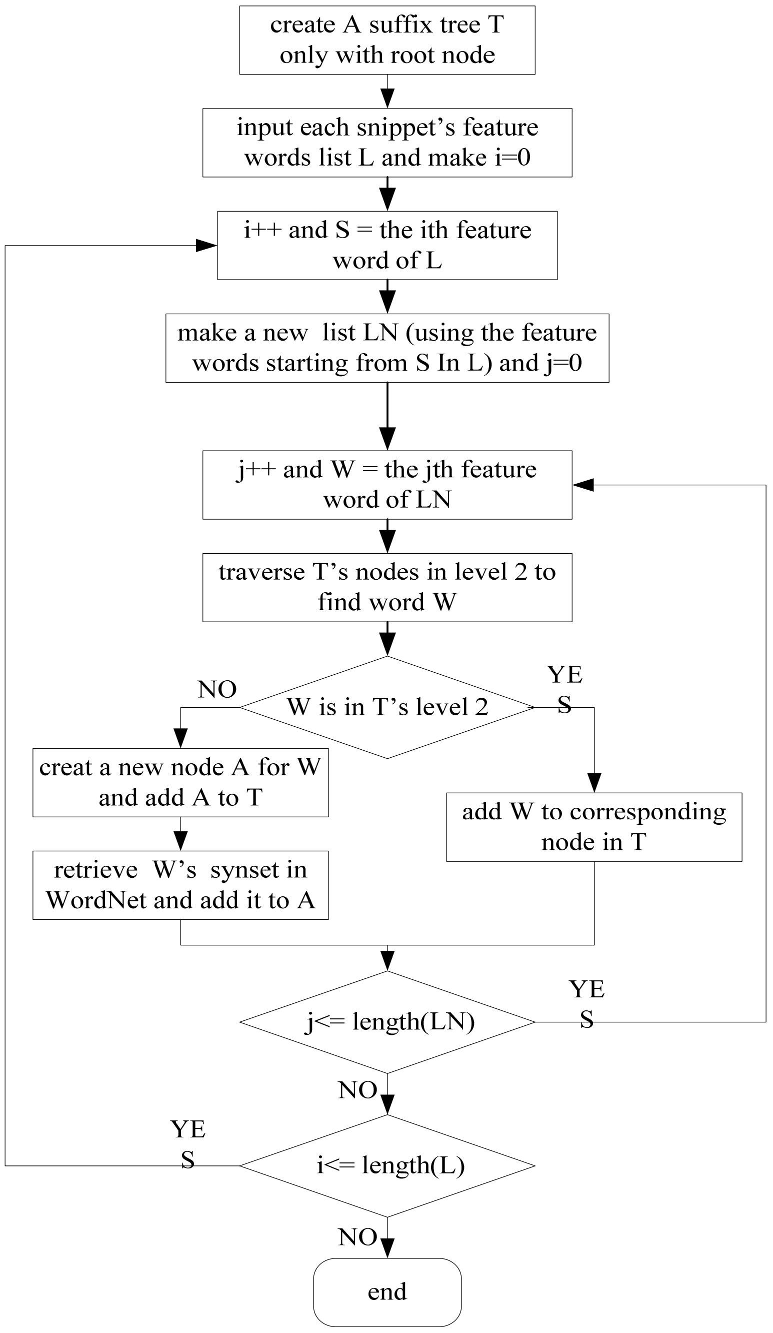

create A suffix tree T only with root node
input each snippet’s feature words list L and make i=0
i++ and S = the ith feature word of L
make a new list LN (using the feature words starting from S In L) and j=0
j++ and W = the jth feature word of LN
traverse T’s nodes in level 2 to find word W
W is in T’s level 2
NO
YES
creat a new node A for W and add A to T
retrieve W’s synset in WordNet and add it to A
add W to corresponding node in T
j<= length(LN)
YES
NO
i<= length(L)
YES
NO
end

Fig. 3 The suffix tree construction flow of WNSTC

There is another example to show the difference between WNSTC’s suffix tree and STC’s suffix tree. Suppose we have two documents: D1 = {car, consume, gasoline}, D2 = {auto, consume, petrol}. Fig. 1 is the conventional suffix tree of STC algorithm. It’s obvious that this suffix tree is sparse and the synonymous words like “car” and “auto” are not recognized. Fig. 2 is the new suffix tree constructed by the method of WNSTC. We can see that the node in Fig. 2 is not a single word any more. Every word contained in WordNet database is combined with its synsets and all these are recorded in corresponding node (within the dotted box). There is curious question: where are "auto" and "petrol"? When constructing the suffix tree, "auto" is found in "car, auto, automobile, machine, motorcar", called node A, so new node for "auto" is no longer created and we link "auto" to node A. The suffix tree in Fig. 2 is compact obviously. We can perceive the semantic advantages of WNSTC to some extent.

The most important difference between WNSTC and STC is reflected when constructing the suffix tree. To keep things simple, we show the suffix tree construction method of WNSTC for one snippet in Fig. 3.

WNSTC Algorithm. WNSTC algorithm has three main steps: preprocessing, constructing suffix tree and identifying clusters. The following sections explain these steps one by one.

Step1: preprocessing. Preprocessing is aim to extract the words or strings from snippets, which have great thematic significance. In linguistics, function words do not have real meaning and cannot act as parts of the sentence generally. Notional words, such as nouns, verbs and adjectives, carry an explicit meaning. And nouns and verbs are the majority of notional words. Consequently, in preprocessing phase of WNSTC algorithm, nouns and verbs are extracted from every snippet to constitute a feature word collection. It's in favor of reducing the nodes number of suffix tree constructed in next step. For example, given a snippet "gasoline is consumed by car", it will be {gasoline, consumed, car} after preprocessing.

Step2: constructing suffix tree. After preprocessing, all snippets are in form of feature words collection. We call the suffix tree construction method of WNSTC described in 2.1 to complete this phase. In order to understand more about the suffix tree construction method of WNSTC, we give an example to show the details constructing suffix tree branch starting from the first feature. We still use the two snippets in section 2.1: D1 = {car, consume, gasoline}, D2 = {auto, consume, petrol}. Consider the diagram a-k in Fig. 4.

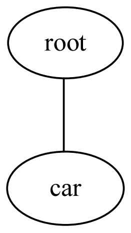
Step3: identifying clusters. Like STC algorithm, there are two parts: identifying base clusters and combining base clusters[3], when identifying clusters. From step 2, the labels under leaf nodes mean a set of documents having common phrase. So, each leaf node which has labels containing more than two documents is a base cluster. And the phrase formed by all nodes' main words on this branch is the topic of this base cluster exactly.

After identifying base clusters, the snippets are grouped to base clusters collections, like {B_1, B_2, B_3, …… , B_n}. If the similarity between two base clusters is 1, they will be combined to one cluster. We use the similarity measure between base clusters in [3]. Given two base clusters B_m , B_n and a parameter α, we define that the similarity between B_m and B_n is 1, if Eq. 1 is satisfied. If not, the similarity between B_m and B_n is 0.

$$\frac{|B_m \cap B_n|}{|B_m|} > \alpha \quad \text{and} \quad \frac{|B_m \cap B_n|}{|B_n|} > \alpha \tag{1}$$

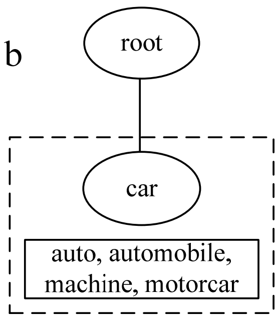
In Eq. 1, | Bm | and | Bn | represent the size of Bm and Bn, respectively. | Bm∩Bn | represents the number of common documents contained in Bm and Bn. Specially, the pre-defined parameter in our method is 0.6, because our experiments using α=0.6 have relatively good performance usually.

a



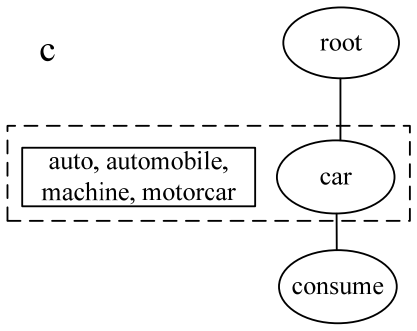
1. creat root node;
2. creat new node for “car” and add it to Tree;

b



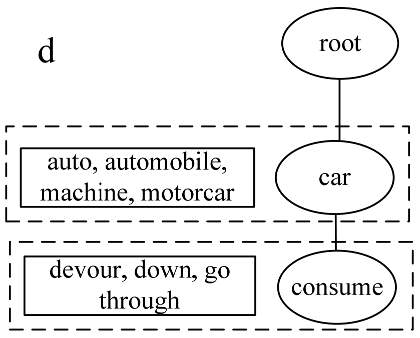
3. ascertain “car” in WordNet;
4. retrieve synsets of “car”;
5. add synsets of “car” to T;

c



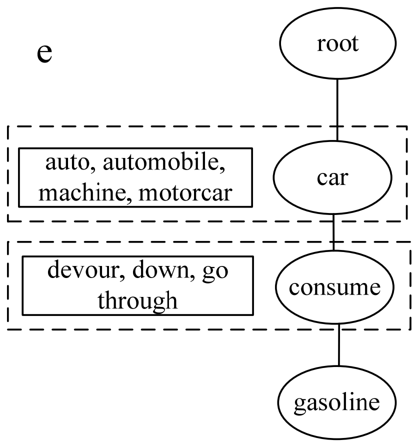
6. creat new node for “consume” and add it to Tree;

d



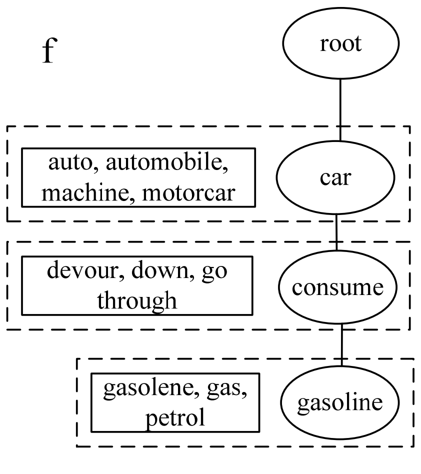
7. ascertain “consume” in WordNet;
8. retrieve synsets of “consume”;
9. add synsets of “consume” to T;

e



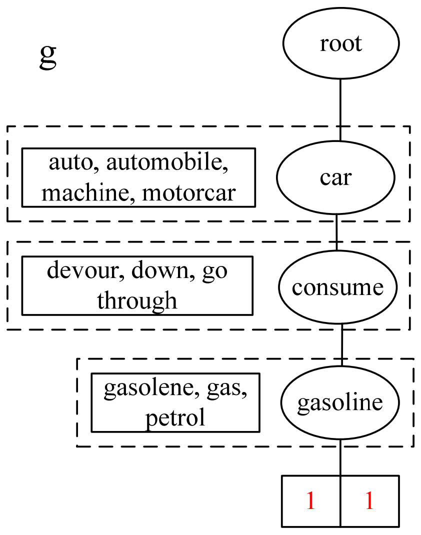
10. creat new node for “gasoline” and add it to Tree;

f

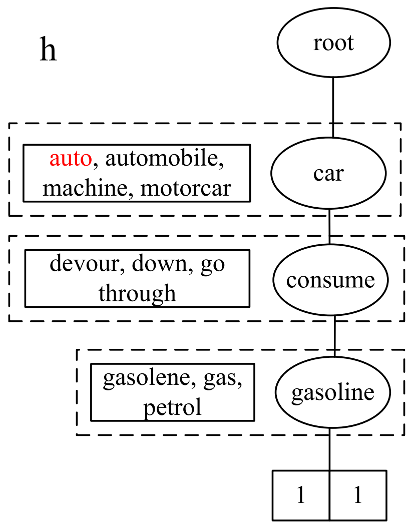


11. ascertain “gasoline” in WordNet;
12. retrieve synsets of “gasoline”;
13. add synsets of “gasoline” to T;

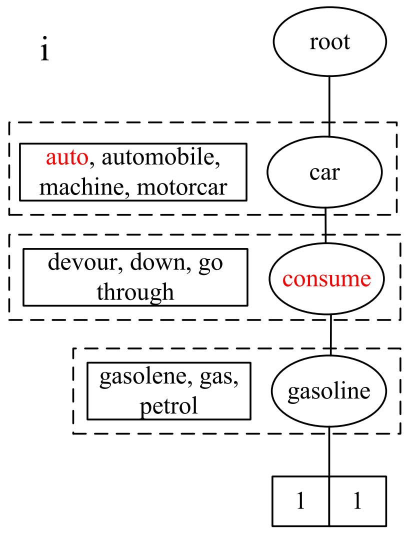
g



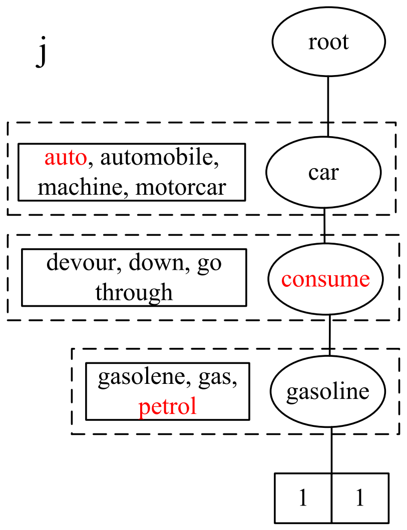
14. creat a label for D1; (The first number means D1, the second number means this branch starts from the first feature word)



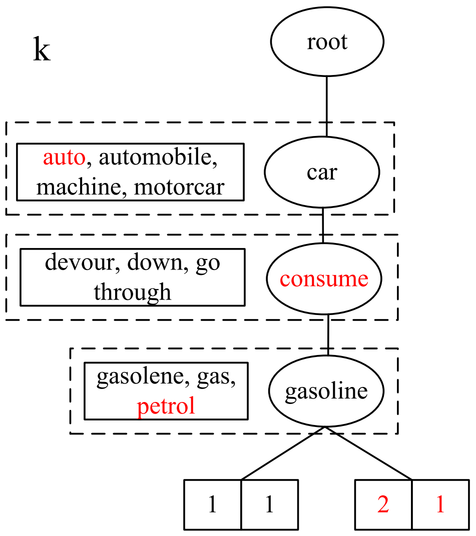
15. “auto” is found in T, link it to T;



16. “consume” is found in T, link it to T;



17. “petrol” is found in T, link it to T;



18. creat other label for D2; (The first number means D2, the second number means this branch starts from the first feature word)

Fig. 4 The details constructing suffix tree branch starting from the first feature

Experiment and Evaluation

Data Sets. In order to get a natural response, we collect experimental data set from Open Directory Project Web – DMOZ [9]. Every website collected in DMOZ has a short introduction, which is very similar to search results' snippet. What's more, DMOZ is the largest, most comprehensive human-edited directory of the Web. It is constructed and maintained by a vast, global community of volunteer editors [10]. For this reason, the classes provided by DMOZ are authentic and objective. Altogether we collect nine data sets. The details of these data sets are showed in Table 1.

Table 1 Data sets for details

Data Set	Totality	Classes (number)
3a	102	shirt (35);swimwear(33); music (34)
3b	73	photography(25);tobacco(30); auto loans(18)
3c	114	music(34);corns(49);Disney(31)
3d	111	drinking games(43);holiday list(33);poultry(35)
3e	86	baseball(20);golf(34);table tennis(32)
3f	135	action movie(14);outdoor organization(102);pizza cooking(19)
4a	136	diamond(37);tobacco(35);photography(33);Disney(31)
4b	88	computer algorithm(25);digital video (24);movie award(23); Biology Genetics(16)
4c	84	action movie(14);newspaper publisher(26);poultry(35);weather(9)

Experiment and Result. We evaluate WNSTC's performance in two ways: measuring clustering accuracy and counting nodes number. There are many evaluation methods for measuring clustering accuracy, such as recall, precision, F-measure, etc. In this paper, we use the classic F-measure to measuring the clustering accuracy of WNSTC. The more accurate clustering is, the higher F-measure value is. Furthermore, we compare WNSTC with STC to demonstrate the advantages of our proposal. Both WNSTC and STC are used to make clustering on every data set in section 4.2. And, the comparison experiments' result is shown in Fig. 5 to Fig. 8.

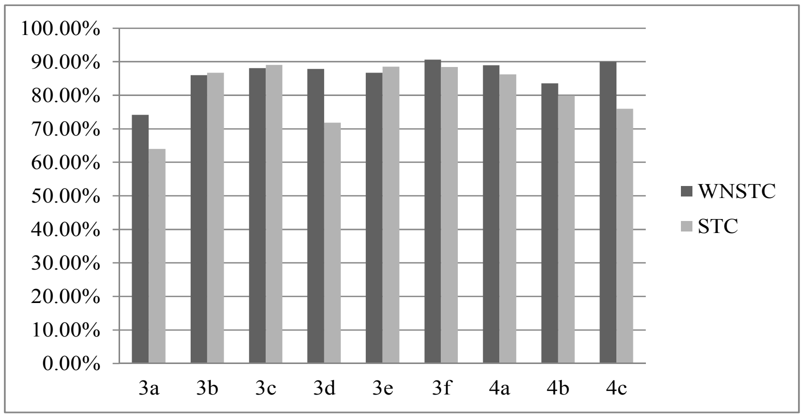


Fig. 5 The F-measure Values of WNSTC and STC

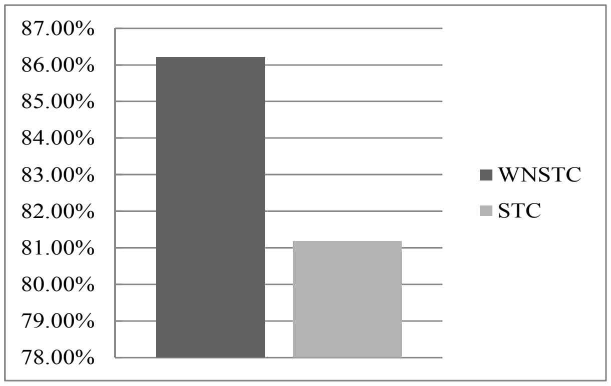


Fig. 6 The Average F-measure Values of WNSTC and STC

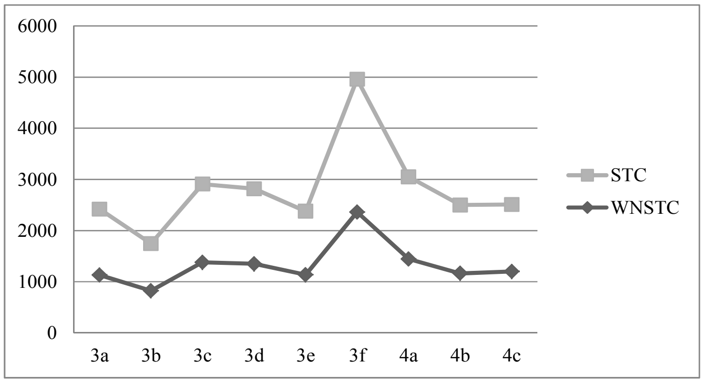


Fig. 7 The Number of Feature Words Comparison

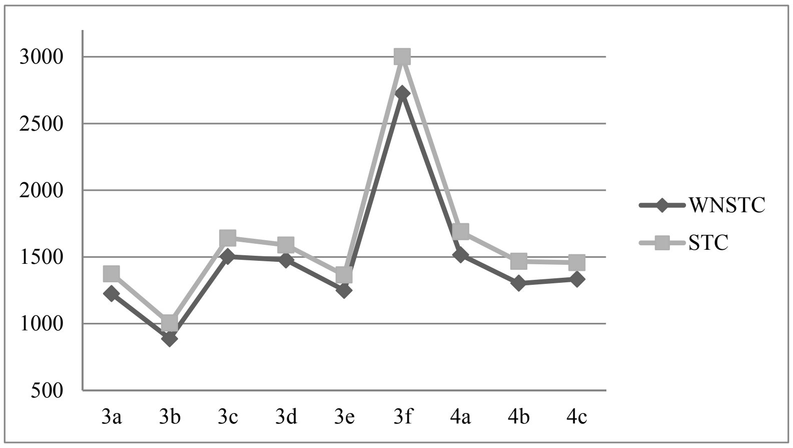


Fig. 8 The Number of Suffix Tree Nodes Comparison

From Fig. 5 the F-measure value of WNSTC is higher than STC or very close to STC. There are six experiment results show that WNSTC has higher F-measure value than STC. Because the dateset itself would have an effect on clustering performance, we calculate the average F-measure for WNSTC and STC. Fig. 6 shows that WNSTC's average performance is always better than STC.

From Fig. 7 and Fig. 8 the feature words and suffix tree nodes of WNSTC are always less than STC. The decreasing percentage we calculated is nearly 10 percent on average. As you know, less suffix tree nodes and less feature words are both good to save storage and improve traversal speed.

Conclusions

This paper has presented a new suffix tree clustering algorithm based on WordNet – WNSTC. WNSTC can recognize the synonyms of snippets' feature word by WordNet and integrate them into suffix tree by expanding the suffix tree nodes. Experimental results show that WNSTC has better clustering quality and smaller suffix tree size than original STC algorithm. However, there are still some points unconsidered in WNSTC which need for further study, such as extracting clustering labels on semantic level and recognizing more semantic information for search results.

References

[1] Krzysztof Strzalka,Aleksander Zgrzywa : Semantic Search Results Clustering. ICCCI 2010, pp. 145–151.

[2] Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, Dawid Weiss: A Survey of Web Clustering Engines, ACM Computing Surveys (CSUR), v.41 n.3, p.1-38, July 2009.

[3] Zamir, O., Etzioni, O.: Web document clustering: a feasibility demonstration. In: SIGIR 1998, pp. 46-54(1998).

[4] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In KDD Workshop on Text Mining, 2000.

[5] Dell, Z., Yisheng, D.: Semantic, Hierarchical, Online Clustering of Web Search Results. In: APWeb 2004, pp.69-78 (2004).

[6] Stanislaw, O., Jerzy, S., A concept-driven algorithm for clustering search results. In: Intelligent Systems, IEEE, Vol. 20, No. 3. , pp. 48-54 (2005).

[7] M.W. Berry, S.T. Dumais, and G.W. O'Brien,Using Linear Algebra for Intelligent Information Retrieval, tech. report UT-CS-94-270,Univ. of Tennessee, 1994.

[8] WordNet, http://wordnet.princeton.edu/.

[9] NAN YANG, YUE LIU, GANG YANG. Clustering of web search results based on combination of links and in-snippets[C]. 2011 Eighth Web Information Systems and Applications Conference. 2011.108-113.

[10] Open Directory Project Web, http://www.dmoz.org/docs/en/about.html.

Acknowledgment

This research was supported by National 863 Program (No. 2011AA01A205), P. R. China.