Suffix Tree Based Chinese Document Feature Extraction and Clustering in RSS Aggregator

Jian Wan, Wenming Yu, and Xianghua Xu
Grid and Services Computing Lab, School of Computer Science and Technology
Hangzhou Dianzi University, Hangzhou 310037, China
1yuwenming_001@163.com

Abstract—In RSS aggregator, the important issue is how to make the feeds information more manageable for RSS subscriber. In this paper, we propose a suffix tree based RSS feeds document clustering in Chinese RSS aggregator. We construct a suffix tree with meaningful Chinese words, and choose the phrases with high score given by a formula as document features. We cluster document using group-average algorithm with a new document similarity measure. The experiment results show that the new method can improve the quality of clustering in document “snippets” scenario, and the speed can meet the demand of “on the fly” clustering.

Index Terms—suffix tree, feature extraction, document clustering, RSS

I. INTRODUCTION

RSS (Really Simple Syndication) is a lightweight XML format designed for sharing headlines and other Web content, and provide news updates from a website in a simple form for web user. RSS feeds benefit readers who want to subscribe the timely updates from favorite websites. However, RSS does not solve the problem of information overload for user, and things tend to get worse as the reader subscribes more and more feeds. One way to deal with the information overload problem is to implement a clustering method within a RSS aggregator. By clustering similar items, a feed aggregator can provide a more friendly interface to user, enable the user to quickly filter duplicate or very similar items [1]. It can also help in filtering out topics that the user is not interested in.

Clustering technique relies on four concepts: data representation model, similarity measure, clustering model and clustering algorithm [2]. From all of these parts, the document representation is the most important, because it determines the way that the other three parts choose. In RSS aggregator, Feed usually send a title and a snippet of content for a feed item along with a link to the full content of that item, so the document representation implemented within RSS aggregators needs to extract the document features from the limited RSS snippet content.

In RSS scenario, we choose suffix tree document model [3] which does not treat a document as a bag of words but rather as a string, making use of proximity information between words. By extracting more information present in the documents, we believe suffix tree document model can help in improving the quality of the clusters.

In this paper, we present a novel method for Chinese snippets clustering. Firstly, we obtain meaningful words (always noun and verb in Chinese) from snippets by Chinese word segmentation at the stage of document preparing. In the construction of Chinese suffix tree, we ignore the nodes (feature phrases) with a high document frequency (df), and only choose the nodes with high score given by a formula we proposed. Then we redefine the pair-wise documents similarity measure for RSS snippet content. With combination of the document features extracted based on suffix tree and the new document similarity, the group-average AHC algorithm is realized in RSS aggregator. The experiment results show that the new method can improve the quality of clustering, and the speed can meet the demand of “on the fly” clustering.

The rest of the paper is presented as follows. Section 2 discusses related work. Section 3 present the improved clustering approach which can be used in Chinese RSS aggregator. Section 4 illustrates some experimental results. Finally Section 5 summarizes our work with some considerations on future directions.

II. RELATED WORK

Document clustering has been investigated as a post-retrieval document browsing technique. Most clustering algorithms base on two document model: the vector space document (VSD) model[4] and the suffix tree document model[3]. Clustering method base on VSD model such as K-Means and agglomerative hierarchical clustering (AHC) cluster the document according to the similarity of vectors which represent documents in the defined vector space. There are several variants from AHC[5], e.g. single-link, group-average, and complete-link. These original algorithms are usually too slow to meet the requirement of “on-line” web applications, such as RSS feeds stream clustering application.

Suffix tree clustering (STC) Algorithm based on Suffix tree document model are usually used in English search’s results clustering[3, 6]. STC is an incremental clustering algorithm and its time complexity is linear with regard to the document corpus size, so it is suitable for clustering web document snippets returned from search engine. However the clustering effectiveness of STC is unsatisfied in Chinese RSS Snippets[7].

Recently, many clustering methods are extended to
specific domains to make the information more manageable. Chim and Deng[8] have proposed a new clustering algorithm combine the advantages of two document models in document clustering. Peng Jing[9] present a novel Chinese text clustering algorithm based on inner product semantic space model. There methods can improve the clustering quality, but they are suitable for “off-line” clustering situation due to time efficiency. Some special methods such as extracting feature code and compressing code[10] are also proposed to solve the clustering problem of short documents.

Compared to aforementioned work, the new clustering algorithm we proposed is to improve the quality of STC in clustering Chinese snippets, and the clustering speed can meet the demand of “on the fly” mode in RSS Aggregator.

III. A NEW CLUSTERING ALGORITHM IN CHINESE SNIPPETS CONTEXT

Our method has three logical steps: (1) document preparing, (2) extracting key phrase using a suffix tree, and (3) clustering snippets using group-average algorithm with a new document similarity measure.

A. Document Preparing

We would ideally like to do the clustering “on the fly” within the RSS aggregator, this means that we do not have time to download the complete content. So we take a title and a snippet of a feed item as a good summary of its content, and treat them as a “document” (document we used in the following means the title and the related snippet) to be clustering. There documents are parsed and split into sentences according to punctuations and HTML tags, and all empty words are stripped. However, different from English document, Chinese words are base units in Chinese document from the view of semantics. Therefore we incorporate Chinese word segmentation into Document Preparing.

Chinese words are fewer than Chinese characters in the same document, and the nodes in suffix tree based on Chinese words can be fewer than on Chinese characters. This means speeding up the construction of suffix tree. Furthermore, the meaningless nodes are removed to improve the accuracy of clustering results. For example, if the Chinese phrase “中山广场” (ZhongShan Square) is identified as a phrase, then its sub-string “广场” (Shan Square) is inevitably selected as a phrase, but this phrase is meaningless [7].

Part-of-Speech (PoS) selection always combines with Chinese word segmentation. In Chinese language, the empty words such as adverb, adjective, preposition, and conjunction act as modifier, and have little power of discrimination. On the other hand, the same semantic snippet can use different empty words based on different context. We can't extract document features effectively if don't remove the empty words which affect identifying of the common phrase. So we only reserve noun and verb for clustering, this can also reduce nodes in a suffix tree.

B. Feature Extraction

We decide to use key phrases extracted from the document collection based on suffix tree as document features. We believe this can help in improving the quality of the clusters by leveraging more information present in the documents.

a. Construct Suffix Tree

The suffix tree data structure was introduced as an efficient string processing technique. A suffix tree allows us to insert a string into the suffix tree incrementally. Following is the definitions related to a suffix tree which was built with Chinese words.

Definition (Suffix of Chinese String): Suppose a string (sentence in document) $d = w_1 w_2 \cdots w_m$ consists of Chinese noun or verb $w_i(i = 1, 2, \cdots m)$, then $S_i = w_i w_{i+1} \cdots w_m$ is a suffix of $d$ starts from the position of $i$.

Definition (Suffix Tree Chinese Document Model): The suffix tree of Chinese documents is a compact tree containing all the suffixes of sentences in documents (designated by leaf nodes). Each edge is labeled with a non-empty substring of sentences. No two edges out of the same node can have edge-labels that begin with the same Chinese word. Each internal node has at least 2 children, represents an overlap phrase shared by at least two suffixes.

Definition (phrase): Phrase is the label of a node, which is designated by the concatenation of the edge-labels on the path from the root to that node.

Figure 1 is an example of a suffix tree composed from 3 simple documents which have processed by document preparing. The 3 processed documents are ‘猫吃老鼠’ (Cat ate mouse), ‘老鼠吃肉’ (Mouse ate meat) and ‘猫 吃 肉’ (Cat ate meat). The nodes of the suffix tree are drawn in circles. Each internal node is attached with a box respectively, each upper number designates a document identifier that presents which document have traversed the corresponding node, each below number designates the phrase occurs in title or in body (0 in body, 1 in title, no title in our 3 example documents).

![Figure 1 The suffix tree of three documents](image-url)
b. Feature Extraction

The suffix tree constructed from documents usually contains lots of (about ten thousand) internal nodes (phrases). Not all internal nodes (phrases) are useful for document clustering, and some of the nodes (phrases) may even misguide the clustering results. So selecting a subset of original nodes as document features can not only reduce the high dimensionality of the feature space, but also improve the accuracy of clustering results. Although the original suffix tree clustering evaluates the quality of nodes with an empirical formula. However, it tends to choose internal nodes containing more documents in our experiment. This means the phrases designated by these nodes have little power of discrimination.

We removed nodes with too high document frequency \( (df) \) using a threshold \( (df - \text{threshold}) \). For example, Many news contains a phrase “记者从相关部门获悉” (The reporter learned from relevant departments that). Although the sub-phrase “记者” (reporter) can express specific meaning in different context, which means we should reserve it. However, we ignore the whole phrase with high \( df \) in the document similarity measure.

Then, we evaluate the phrase importance by statistical method which is usually applied in VSD model. For each internal node \( n_i \), the phrase designated by \( n_i \) is \( p_i \). When we are constructing the suffix tree, use variable \( tf(p_i) \) accumulate the times traverse through the node \( n_i \) by the suffix in the documents corpus, then \( tf(p_i) \) is the term frequency of phrase \( p_i \); the times of different documents that traverse through the node \( n_i \) is \( df(p_i) \), then \( df(p_i) \) is the document frequency. Therefore the weight of phrase \( p_i \) in documents corpus can be calculated using the classic \( tf/ idf \) scheme in formula (1), where \( N \) is the total number of documents in corpus.

\[
\text{tf \_idf} = \log(tf(p_i)) \cdot \log(N / df(p_i)) \tag{1}
\]

A phrase is an ordered sequence of one or more words. The more number of words a phrase contains, the richer meaning it can express. Therefore, the importance of a phrase should incorporate a factor about the length of phrase \( p_i \), which designated by \( |p_i| \). We calculate the factor using a heuristic utility function in following formula:

\[
f(|p_i|) = \log_2 |p_i| \tag{2}
\]

The score \( s(n_i) \) of node \( n_i \) (phrase \( p_i \)) is given by formula (3). To speed up the follow clustering, we only choose the \( k \) highest scoring phrases as key phrases (we take \( k \) to be 1000 in our experiment).

\[
s(n_i) = tf \_idf \cdot f(|p_i|) \tag{3}
\]

C. Document Clustering

The document clustering approach we presented is an improved group-average algorithm[5] in which the pair-wise documents similarity measure is modified according to RSS context. The advantage of group-average clustering algorithm in our application is it can be stopped at any point when the remaining pairs of clusters have low similarity values, so it doesn’t have the problem of selecting appropriate initial number of clusters. Most importantly, group-average algorithm can always achieve better clustering result than other algorithms.

Group-average algorithm is argued for spending too much time in clustering. But in our clustering problem, the number of clusters \( k \) tends to be comparable to the number of documents \( n \). This is because there are few similar news about the same topic, majority of clusters have only several news each. Therefore, the group-average algorithm with time efficiency \( O(n^2) \) can be faster than EM or K-Means implementations, which is \( O(knf) \) (where \( f \) is the number of features per document) per iteration.

We redefine the pair-wise documents similarity abandoning the cosine document similarity measure to speed up the Clustering. From the suffix tree constructing, it’s very easy to understand that the more internal nodes shared by two documents, the more similar the documents tend to be. Since the key phrase in the title of a document is more representative than in the body, it is reasonable to distinguish different situation with different weight. So we use the doc_similarity algorithm (shown in Figure 2) to measure the similarity between two documents.

1: void doc_similarity(sorted array doc1, sorted array doc2) {
2:     int i = 0;
3:     int j = 0;
4:     int similarity = 0;
5:     while ((i<k) && (j<k)) {
6:         if(doc1[i] == doc2[j]) {
7:             if(doc1[i].pos==1 && doc2[j].pos==1) {
8:                 similarity += 5;
9:             } else if(doc1[i].pos==0 && doc2[j].pos==0) {
10:                similarity += 3;
11:            } else {
12:                similarity += 1;
13:            }
14:         } else if(doc1[i]<doc2[j]) {
15:             i++;
16:         } else if(doc1[i]>doc2[j]) {
17:             j++;
18:         }
19:     }
20: }

Figure 2 The function to measure the similarity between two documents
Each document can be expressed with an array related to the $k$ key phrases if we traverse the suffix tree once. We sort each array to make it ordered before measuring the similarity.

IV. EVALUATION

In this section, we evaluate the effectiveness and efficiency of our algorithm. The algorithms to be compared are the original STC and group-average algorithm with the traditional term $tf/\text{idf}$ cosine similarity measure. We use JAVA as the tool for simulation, and our experiment equipment is a PC with Pentium(R) 4 CPU 3.00GHz, 1024 MB memory and MS Windows XP operating system.

A. Dataset

Google News service (http://news.google.cn/) uses document clustering techniques to group news articles from multiple news sources, it provide an easily available labeled dataset for evaluating our clustering results.

Each article in Google News homepage has a link pointing to the other articles on the same topic. We choose 8 articles from different topics (e.g. business, sports, health, Entertainment, etc.) and collect the top 100 related articles for each topic. This way we gathered 800 new articles belonging to the 8 different topics. Each news article consists of a title and a description snippet of content.

B. Quality measure

We use commonly used $F$-measure for evaluating and comparing different clustering results. $F$-measure combines the Precision and Recall ideas from the Information Retrieval literature. The precision and recall of a cluster $j$ with respect to a “correct” class $i$ are defined as:

$$P = \text{Precision}(i, j) = \frac{N_i}{N_j}$$ (4)

$$R = \text{Recall}(i, j) = \frac{N_i}{N_i}$$ (5)

where

$N_i$: is the number of members of “correct” class $i$ in cluster $j$,
$N_j$: is the number of members of cluster $j$, and
$N_i$: is the number of members of “correct” class $i$.

The $F$-measure of a class $i$ is defined as:

$$F(i) = \frac{2PR}{P+R}$$ (6)

C. Results

Although group-average can be easy to tune the similarity threshold to get appropriate clusters in real RSS aggregator, we select 8 initial clusters beforehand to achieve a fair result.

Figure 3 illustrates the $F$-measure scores computed from three clustering algorithms on the dataset we get above. NSTC designates the algorithm we describe above, and GTC designates group-average algorithm with traditional term $tf/\text{idf}$ cosine similarity measure.

As shown in Figure 3, the performance of NSTC is improved compared with the two other algorithms. This is mainly due to three facts: (1) We construct the suffix tree based on Chinese words rather than Chinese characters. (2) We extract high discrimination phrases from documents, and remove the meaningless phrases which misguide the clustering result. (3) Our new document similarity measure has the ability to accurately judging the relation between snippets.

Figure 3 $F$-measure of the three clustering algorithms results

Speed also plays an important role in the implement of clustering for RSS aggregator. We measure the execution time of the different algorithms while clustering snippet collections of various size (100~800 snippets, evenly distributed in 8 different topic as far as possible). The results are shown in Figure 4.

Figure 4 Execution time (in seconds) of different algorithms while clustering snippet collections of various sizes

Although NSTC is much faster than the GTC, it is slower than STC algorithm. The main reason is that group-average algorithm wastes a little time. However, the number of snippets to clustering in RSS aggregator is no more than 400 in most cases. So the efficiency can meet the demand of on-line interaction in RSS context.

V. CONCLUSION AND FUTURE WORK

In this paper, a new document clustering algorithm is introduced for the Chinese snippets in RSS aggregator. We extract effective phrases from snippets based on
suffix tree, and clustering RSS snippets using group-average algorithm with a new document similarity measure. The experiment results show that the new method can improve the quality of clustering, and the speed can meet the demand of “on the fly” clustering in RSS aggregator. In future work, we intend to introduce world knowledge such as HowNet (An electronic dictionary of Chinese like WordNet) to improve the effectiveness of our short Chinese document clustering method, and applying this snippets clustering into other domains such as email clustering.

ACKNOWLEDGEMENTS

This paper is supported by National Science Foundation of China under grant No.60873023, and Science and Technology R&D Program of Zhejiang Province, China under grant No. 2008C13080, No.2007C21G3230005.

We thank Dawid Weiss and Stanislaw Osinski for their contributions to open source framework of clustering (http://project.carrot2.org/), by referring which we reduce amount of time for our research. Thanks also to Institute of Computing Technology, Chinese Academy of Sciences for their excellent work in the implementation of Chinese word segmentation (http://ictclas.org/). We make use of ICTCLAS module in the stage of document preparing.

REFERENCES


