Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 723469, 9 pages
http://dx.doi.org/10.1155/2015/723469

Research Article

Text Matching and Categorization: Mining Implicit Semantic Knowledge from Tree-Shape Structures

Lin Guo,[1,2] Wanli Zuo,[1,2] Tao Peng,[1,2] and Lin Yue[1,2]

[1]*College of Computer Science and Technology, Jilin University, Jilin 130000, China*
[2]*Symbol Computation and Knowledge Engineer of Ministry of Education, Jilin University, Jilin 130000, China*

Correspondence should be addressed to Wanli Zuo; wanlizuo@126.com

Received 31 March 2015; Accepted 9 June 2015

Academic Editor: Chaudry Masood Khalique

The diversities of large-scale semistructured data make the extraction of implicit semantic information have enormous difficulties. This paper proposes an automatic and unsupervised method of text categorization, in which tree-shape structures are used to represent semantic knowledge and to explore implicit information by mining hidden structures without cumbersome lexical analysis. Mining implicit frequent structures in trees can discover both direct and indirect semantic relations, which largely enhances the accuracy of matching and classifying texts. The experimental results show that the proposed algorithm remarkably reduces the time and effort spent in training and classifying, which outperforms established competitors in correctness and effectiveness.

1. Introduction

Rapid developmental trend in social network means the explosive growth of users as well as dramatic changes in providing services. Therefore, large-scale text classification and retrieval revive the interest of researchers [1]. The traditional knowledge representations are characterized by strong pertinences and have great power in expressing empirical knowledge or rules, but they are insufficient in representing complex and uncertain knowledge existent in social webs. Texts share various forms of common structural components (from simple nodes and edges to paths [2, 3], subtrees [4], and summaries [5]) [6]. Direct semantic information can be found easily, but hidden semantic information is extremely difficult to be detected. Zaki and Aggarwal [4] propose a structural rule-based classifier for semistructured data, called XMiner, which can mine out parent-child frequent branches and ancestor-descendant ones and conduct structured or semistructured data perfectly, but the shortness is the lack of semantic information in text representation.

Semantic similarity assessment [7, 8] can be exploited to improve the accuracy of current information retrieval techniques [9], to automatically annotate documents [10, 11], to protect privacy [12, 13], to match web services [14], and to resolve problems based on knowledge reuse [15]. Semantic network [16–18] is more concerned about semantic information. For the semantic data mining can be based on the text analysis, many semantic community detection algorithms exploited the latent Dirichlet allocation (LDA) model as the core model, which is a generative model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar [19, 20]. However, semantic analyzing based on LDA [16, 21] is complicated, and semantic information mining is important for text matching and categorizing, so it is needed to find a much more efficient and friendly way, of which the results are precise and accurate.

A relation between two words can be in one-way direction or bidirection based on the interrelationships between them, so it is reasonable to use graphs or trees to express a text. The method proposed can mine out implicit semantic information without cumbersome lexical analysis by making links express semantic knowledge and pointers record a traversal sequence which describes different abilities of nodes in expressing a text. The method proposed in this paper not only extracts semantic information by creating tresses but also calculates the similarities of coexisting hidden structures to measure the similarities of texts. Three main contributions of

Input: text T
Output: SemGraph G
(1) scan sentence n
(2) IF sentence n is not empty THEN
(3) create $L_n = \{w_1, w_2, \ldots, w_m\}$
(4) ELSE exist
(5) FOREACH w_x IN L_n ($x = 1, 2, 3, \ldots, m$) DO
(6) IF G = NIL THEN
(7) add w_x to G
(8) $Count(w_x) = 1$
(9) ElSE
(10) IF w_x does not appear in G THEN
(11) add w_x to G
(12) add all edges of w_x to G
(13) $Count(w_x) = 1$
(14) IF w_x appears in G THEN
(15) $Count(w_x) = Count(w_x) + 1$
(16) FOREACH two linked nodes IN G DO
(17) IF the *Counts* of them are equal THEN
(18) set the direction of the pointer randomly
(19) IF the *Counts* of them are unequal THEN
(20) set the direction of the pointer from the node with a bigger *Count* to the one with a smaller *Count*

Algorithm 1: ESemGraph().

this work are listed as follows. One is to represent all semantic information in a text using tree-shape structures. The other is to generate semantic trees based on the combining of pointers and a fixed traversal strategy and to use subtrees as addenda structures. The last one is to discover implicit knowledge by analyzing semantic trees and mining coexisting hidden structures.

2. Representation of Semantic Information

Because knowledge model is highly dependent on relations, it is reasonable to use trees to express a text. This paper employs tree-shape structures to describe a text, from which semantic information can be mined out without cumbersome lexical analysis.

2.1. Semantic Graphs. A text is deemed as a sequence of sentences (denoted as $T = \langle L_1, L_2, L_3, \ldots \rangle$, where L_n is the nth sentence in text T), and a sentence is viewed as a sequence of words (denoted as $L_n = \langle w_1, w_2, w_3, \ldots \rangle$, where w_m is the mth noun in sentence L_n). Nouns are extracted from texts to create L_n, because nouns are capable of describing the meaning of a sentence.

Definition 1 (SemGraph). It is a semantic graph and is denoted as $\langle N, E, P \rangle$, where N is a set of nodes and each contains the information about the frequency and weight of a noun, E is a collection of edges representing relations among nodes, and P is a collection of pointers used as the guidelines when traversing a graph, which depicts the descending order of abilities of nodes in N.

Based on the assumption that words in one sentence are deemed as having semantic relationships (a relationship is existent between w_i and w_j in L_n, where $i, j \in [1, n]$ and $i \neq j$), the nodes arising in the same sentence are linked with each other in SemGraph.

Definition 2 (isolated node). The node has neither in-degree pointers nor out-degree pointers.

The process of building SemGraph is as in Algorithm 1.

The creation of a SemGraph is illustrated by the example shown in Figure 1. By scanning a text in sentences and supposing nouns C_1 and C_2 appear in the first sentence, they are added into SemGraph directly and the *Counts* of them are assigned to 1, respectively (Figure 1(a)). Since the *Counts* of the two nodes are equal, the direction of the pointer is set randomly. Figure 1(b) supposes C_1 and C_3 coexist in the second sentence. Because C_1 has existed in SemGraph, there is no need to add C_1 to the SemGraph again, but the *Count* of C_1 must be modified ($Count = Count + 1$). As a new node, C_3 is added to the SemGraph directly and the *Count* is set to 1. Because $Count(C_1) = 2 > Count(C_2) = 1$, the direction of the pointer is shifted from C_1 to C_2. Similarly, for $Count(C_1) = 2 > Count(C_3) = 1$, the pointer between them is set from C_1 to C_3. In short, pointers mark from the more frequent node to the opposite. In Figure 1(c), sentences in the text are supposed to be as follows: $\langle C_4\ C_5 \rangle$, $\langle C_4\ C_6\ C_7 \rangle$, $\langle C_4\ C_6 \rangle$, $\langle C_2\ C_4 \rangle$, $\langle C_1\ C_5 \rangle$, $\langle C_3\ C_7 \rangle$, $\langle C_6\ C_3 \rangle$. After processing the similar works mentioned above for each sentence, the final result is shown in Figure 1(c). If the last sentence is $\langle C_6\ C_3 \rangle$, the *Counts* of C_3 and C_6 add one, respectively. Because $Count(C_3) = 4 > Count(C_1) = 3$ and $Count(C_6) = 4 > Count(C_4) = 3$, the pointers between C_1-C_3 and C_4-C_6 should be changed, as shown in Figure 1(d).

After building an original SemGraph, the redundant nodes (the *Counts* under threshold K) must be pruned to

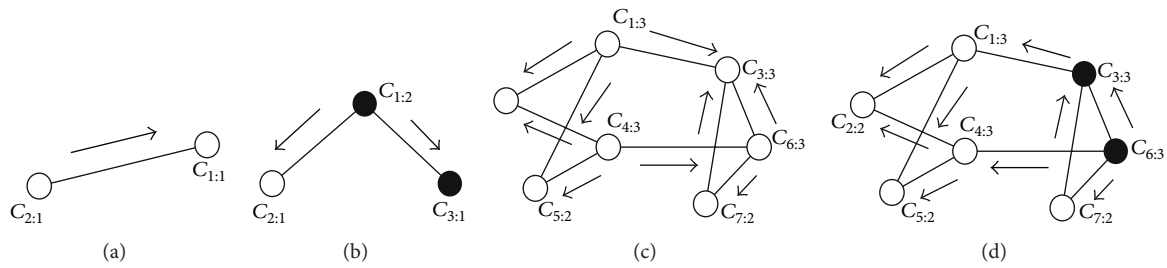

Figure 1: The creation of a SemGraph. Black symbols represent the items changed. Nodes are denoted as [node name: *Count*].

achieve the purpose of simplification as they are weak to describe a text.

The following is the setting method of threshold K:

$$S_c = \sum_0^n Count\,(n_i)\,,$$

$$S_n = \mathrm{num}\,(G)\,,$$

$$\varepsilon = \left\lfloor \frac{S_c}{S_n} \right\rfloor \times T \times \frac{1}{e_v}, \tag{1}$$

$$\mathrm{K} = \frac{1}{1 + \omega \exp\,(-\varepsilon)},$$

$$\omega \in (0, 1]\,.$$

S_c is the sum of *Counts*. S_n is the number of nodes. T is the number of characters in the text. e_v is the average number of characters in samples. ω is an optional artificial setting value. ε is a measurement of SemGraph by considering various parameters, and K controls ε in the Scope of (0, 1). If some texts are more authoritative or have stronger abilities to represent a class, ω is reset to a smaller value based on specialist knowledge. The smaller the ω is, the more important the text is. Eventually a smaller K makes more information in the text retained.

Further explanations are as follows. (1) K is inversely proportional to the mean of *Counts* in SemGraph. (2) More characters in a text lead to more redundant information, so the size of the text is used to fine tune ε. (3) Experts can manually select some representative texts and assign a smaller ω for building a SemGraph representing a class (denoted as class-SemGraph) quickly.

2.2. Fusion Strategy. SemGraphs that represent texts are denoted as text-SemGraph. A class-SemGraph is generated by merging text-SemGraphs in the same class. The following two problems need to be considered to merge different text-SemGraphs in the same class. (i) Weights of nodes must be recalculated. The formula is as follows: $W = Count(n_i)/\sum Count(n_k)$, where n represents a node and W embodies the importance of node n_i in the corresponding class. (ii) If the number of occurrences of a new relationship is larger than threshold Δ_{addedge}, the link between the two nodes will be added in the class-SemGraph. If the number of disappearances of an old relation is less than threshold

Input: new added SemGraph *newG*; *time* = 0
Output: class-SemGraph *G*
(1) IF *newG* = NIL THEN exit
(2) IF *G* = NIL THEN *G* = *newG*
(3) ELSE
(4) unify names in *G* and *newG*
(5) calculate *W* for each node
(6) calculate the sum of *W* of same nodes
(7) FOR $a \in newG$ && $b \in newG$ && $a \neq b$ DO
(8) IF $E_G(a, b) = 1$ && $E_{newG}(a, b) = 1$ THEN
(9) $\omega_{ab} = \omega_{ab} + 1$
(10) IF $E_G(a, b) = 0$ && $E_{newG}(a, b) = 1$ THEN
(11) $\omega_{ab} = \omega_{ab} + 1$
(12) IF $\omega_{ab} > \Delta_{\text{addedge}}$ THEN create the link
(13) IF $E_G(a, b) = 1$ && $E_{newG}(a, b) = 0$ THEN
(14) $\omega_{ab} = \omega_{ab} - 1$
(15) IF $\omega_{ab} < \Delta_{\text{subedge}}$ THEN delete the link
(16) delete *a* and *b* in *newG*
(17) deleNode(*G*, *time*)
(18) *time* = *time* + 3

Algorithm 2: MSGraph().

Δ_{subedge}, it means the relationship is weak or nonexistent, so it should be deleted in the class-SemGraph.

By adding new classified texts to the corresponding class-SemGraph, high accuracy and real-time performance can be ensured. In conclusion, the merging operation is needed to be performed by combining text-SemGraphs for creating or updating class-SemGraphs. The implement strategy is as in Algorithm 2.

$E_u(p, q)$ judges whether there is a link between node p and q in SemGraph u. $E_u(p, q) = 1$, if a link exists between the nodes and 0 otherwise.

Insignificant nodes in G must be deleted every once in a while to ensure timeliness, which is done by Algorithm 3. Nodes deemed as less capable to describe a class must satisfy the following conditions: $N_b(node_i) < \sigma_n$ && $Count(node_i) < \sigma_c$, where $N_b(node_i)$ is the sum of in-degree and out-degree pointers of the ith node and σ_n is an artificial threshold and proportional to the average length of texts. The root of G should be relocated when the network is changed, which is the start node when traversing G or mining frequent structures. The concrete implementation of finding a root is also done by Algorithm 3.

Input: SemGraph G; the time interval parameter t
Output: a simplified SemGraph G; the root node R
(1) IF t meets the restriction of time THEN exit
(2) ELSE
(3) FOREACH $node_i$ IN G
(4) IF $N_b(node_i) < \sigma_n$ && $Count(node_i) < \sigma_c$ THEN
(5) delete $node_i$ and all edges of $node_i$
(6) R records the node that has the maximum $Count$
(7) return R

Algorithm 3: deleNode(G, t).

2.3. Formation of Trees. In order to analyze implicit frequent structures, SemGraphs should be decomposed into several trees; thus studying the features of the trees is equivalent to processing the SemGraph. Depth-First Search (DFS) or Breadth-First Search (BFS) strategies cannot meet the requirements of social network, because fixed traversal strategies would miss or destroy some important relationships; thus pointers are needed to achieve correct mining results when traversing graphs. The method of choosing a root is as follows: (1) choose the node with the maximum *Count*; (2) if there is more than one node having equivalent maximum *Count*, the node having more out-degree pointers is chosen as the root.

BuildTree() is a semantic graph searching method proposed in this paper without losing semantic information based on DFS or BFS. BuildTree() usually generates more than one tree, so several trees can express all semantic relationships between nodes. Algorithm 4 is the semantic graph searching strategy based on DFS.

In Figure 1, C_6 has the biggest *Count*, so C_6 is chosen to be the root. BuildTree() creates three sets of trees based on DFS and BFS, respectively, shown in Figures 2 and 3. The analysis shows that in spite of the two different results they do not affect follow-up works as they express exactly the same semantic information.

T_0 is a master subtree, while both T_1 and T_2 are auxiliary subtrees. DFS or BFS only create master subtrees, which omit some vital semantic relationships. For instance, T_0 believes that C_2 and C_4 have no semantic relation, but actually they have one in SemGraph, so T_1 is essential to replenish this missing relationship.

Input: SemGraph G; the root node R; int $n = 0$
Output: trees (denoted as t_n, $n = \{1, 2, 3, \ldots\}$)
(1) IF R is empty THEN
(2) select R from G
(3) $n = n + 1$
(4) add R to t_n
(5) visit all subsequent nodes of R according to the pointers
(6) LN records all the subsequent nodes
(7) IF LN is empty THEN return
(8) ELSE
(9) FOREACH $node$ IN LN
(10) link R with $node$ in t_n
(11) delete the edges that have been visited in G
(12) IF $node$ is an isolate node THEN delete $node$
(13) ELSE BuildTree($node$)

Algorithm 4: BuildTree(R).

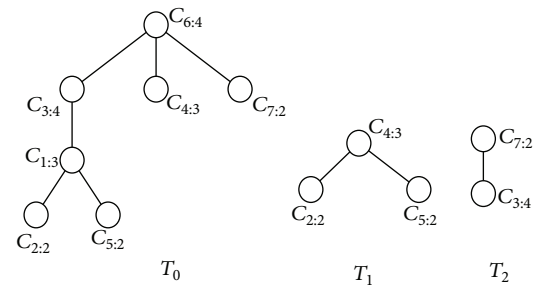

Figure 2: The result of semantic graph searching strategy based on DFS.

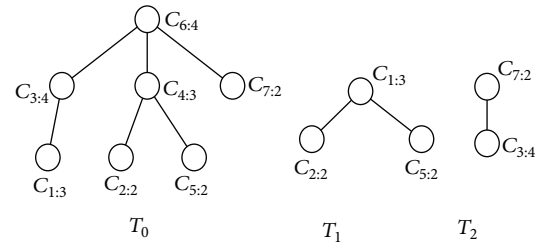

Figure 3: The result of semantic graph searching strategy based on BFS.

3. Mining Implicit Frequent Structures

Definition 3 (implicit frequent structure (IStruc)). IStruc is a frequent structure of SemGraph, which reserves ancestor-descendant relationships.

That is, there are at least two connected nodes in IStruc and they are not linked in SemGraph; the frequent structure like this is called implicit frequent structure.

Definition 4 (Scope). It represents the Scope of node N in a tree, whose format is $[L_x, L_y]$, where L_x is an index of node N generated by traversing the tree according to DFS or BFS and L_y is the maximum value of L_x's among all successor nodes of N.

Definition 5 (branch root). It meets the following conditions: L_x is the smallest one in all L_x's and L_y is the smallest one in all L_y's.

Definition 6 (List). It is denoted as $[F_n, T_n$, Scope, (Scope of branch root)$^{Ik}]$, where F_n is the ID of a text, T_n is the ID of a tree, and Ik is the number of branch nodes.

In order to mine IStrucs, it is needed to analyze new structures generated by connecting nodes one by one. But there is no need to connect all the nodes. For example, if two nodes in a tree do not have a common ancestor node, they should not be connected. Therefore, it is essential to judge whether the nodes meet some preconditions.

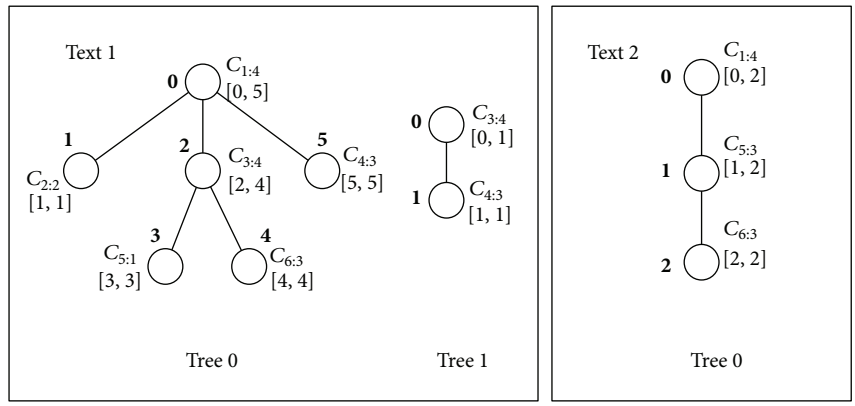

FIGURE 4: Two sets of trees representing different texts. Text 1 is expressed by two trees, while Text 2 is explained by one tree. The bold numbers represent the node IDs generated by DFS.

Preconditions. If node A (the List is $\langle F_a = F_b,\ T_a = T_b,\ [L_{xa}, L_{ya}],\ \{[L'_{xa}, L'_{ya}]\}^{Ik}\rangle$) and node B (the List is $\langle F_b = F_a,\ T_b = T_a,\ [L_{xb}, L_{yb}],\ \{[L'_{xb}, L'_{yb}]\}^{Ik}\rangle$) are linked in a IStruc, they must meet one of the following conditions. (1) If $L_{xa} \geq L_{xb}$ && $L_{ya} \leq L_{yb}$, A is a child node of B in the SemGraph. (2) If $L_{ya} \leq L_{xb}$ && A and B have the same ancestor node, A is a brother node of B in the SemGraph. (3) If $L'_{xa} \geq L'_{xb}$ && $L'_{ya} \leq L'_{yb}$ && $L_{xa} = L_{xb}$ && $L_{ya} = L_{yb}$, A is a child node of B's branch root in the SemGraph. (4) If $L'_{ya} \leq L'_{xb}$ && $L_{xa} = L_{xb}$ && $L_{ya} = L_{yb}$, A is a brother node of B's branch root in the SemGraph.

To specify the process of mining IStrucs by computing the Scopes of nodes, two sets of trees representing two texts are shown in Figure 4. The subscript of C_1 in Tree 0 of Text 1 is 0 determined by DFS, so $L_x = 0$. All the direct successor nodes of C_1 are $\{C_2, C_3, C_4\}$, and L_y's of those nodes are $\{1, 4, 5\}$. Obviously, C_4 has the maximum L_y ($L_y = 5$), so the L_y of C_1 is set to 5 and the Scope of C_1 is $[0, 5]$.

The final Lists are shown in (2). There are six different nodes in the tree, so six Lists are set up. Taking C_1 as an example, C_1 appears in Tree 0 of Text 1 and Tree 0 of Text 2, and the Scopes are $[0, 5]$ and $[0, 2]$, so the List of C_1 has two items, which are $\langle 1, 0, [0, 5]\rangle$ and $\langle 2, 0, [0, 2]\rangle$.

Lists of nodes are as follows. The format of an item in a list is ⟨text ID, tree ID, Scope⟩:

C_1	C_2	C_3	C_4	C_5	C_6
1, 0, [0, 5]	1, 0, [1, 1]	1, 0, [2, 4]	1, 0, [5, 5]	1, 0, [3, 3]	1, 0, [4, 4]
2, 0, [0, 2]		1, 1, [0, 1]	1, 1, [1, 1]	2, 0, [1, 2]	2, 0, [2, 2]

(2)

The node just appearing in one text cannot be a common IStruc, so nodes like this are deleted. C_2, C_3, and C_4 only appear in Text 1, so they are deleted. After deleting redundant nodes, the rest are $\{C_1, C_5, C_6\}$. Assuming that C_1 is a root node, it will be linked with other nodes which meet the preconditions to create new IStrucs. Therefore, $\{C_1\ C_5\}$ and $\{C_1\ C_6\}$ are created as that shown in (3).

Lists of structures are as follows. The format is ⟨text ID, tree ID, the Scope of the root, the Scope of branch root⟩:

$\{C_1\ C_5\}$	$\{C_1\ C_6\}$
1, 0, [0, 5] , {[3, 3]}	1, 0, [0, 5] , {[4, 4]}
2, 0, [0, 2] , {[1, 2]}	2, 0, [0, 2] , {[2, 2]}

(3)

The interpretation of the result $\langle 1, 0, [0, 5], \{[3, 3]\}\rangle$ generated by $\{C_1\}$ $\{C_5\}$ is as follows: After visiting the Lists of C_1 and C_5, it is obvious that $\langle 1, 0, [0, 5]\rangle$ and $\langle 1, 0, [3, 3]\rangle$ meet the preconditions (text IDs of the two nodes are 1; tree IDs are 0; $\{L_{x,c1} = 0 \leq L_{x,c5} = 3\}$ && $\{L_{y,c5} = 3 \leq L_{y,c1} = 5\}$). Therefore, $\{C_1\}$ $\{C_5\}$ are in a father-child relationship and C_1 is a branch root, because the Scope of C_1 is wider than that of C_5. Finally, the List of $\{C_1\ C_5\}$ is $\langle 1, 0, [0, 5], \{[3, 3]\}\rangle$. After getting the Lists of $\{C_1\ C_5\}$ and $\{C_1\ C_6\}$, it can be discerned that they also meet the preconditions, so structure $\{C_1\ C_5\ C_6\}$ is calculated on the basis of $\{C_1\ C_5\}$ and $\{C_1\ C_6\}$, as shown in (4).

The result of $\{C_1\ C_5\ C_6\}$ is as follows:

$\{C_1\ C_5\ C_6\}$
1, 0, [0, 5] , {[3, 3]} , {[4, 4]}
2, 0, [0, 2] , {[2, 2]}

(4)

After several linking operations, two results are gained. $\langle 1, 0, [0, 5], \{[3, 3], [4, 4]\}\rangle$ is the structure shown in

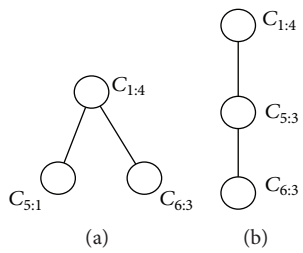

Figure 5: IStruc.

Figure 5(a) existing in Tree 0 of Text 1. $\langle 2, 0, [0, 2], \{[2, 2]\}\rangle$ means that Tree 0 of Text 2 has the structure shown in Figure 5(b). Nodes $\{C_5\ C_1\ C_6\}$ are not directly connected in the original trees, so C_5-C_1-C_6 is an implicit frequent structure. Although the mining results have different structures in SemGraphs, they contain the same hidden knowledge. Without mining IStrucs, some implicit relations of texts are ignored entirely, which greatly reduces the accuracy of text matching.

4. Scoring Tactics

The semantic trees having common IStrucs are not a proof of existing association relations, so it is essential to analyze the authorities of IStrucs. The following is the scoring tactic of IStrucs to compute similarities between two texts or between an unknown-class text and a class.

Scoring rules:

$$\delta = \frac{\sum Count\left(n_i\right)}{\sum Count\left(m_j\right)}, \qquad \Delta = \left|\delta_{\text{new}} - \delta_{\text{class}}\right|, \tag{5}$$

where n is a node in an IStruc, while m is a node in a SemGraph. Δ is the degree of variance between a text and a class.

From the above example, the differences between Text 1 and Text 2 can be computed. The score of the IStruc in Tree 0 of Text 1 is $\delta_1 = (4 + 1 + 3)/(4 + 2 + 4 + 3 + 1 + 3 + 4 + 3) = 0.3333$. The score of the IStruc in Tree 0 of Text 2 is $\delta_2 = (4 + 3 + 3)/(4 + 3 + 3) = 1$. So the distance of the two texts is $\Delta = |1 - 0.3333| = 0.6667$.

5. Experiment

Three datasets are used in this paper.

(1) SND: the dataset is gathered from sina (http://www.sina.com/) repeatedly. Training data is collected in different periods, and testing set dynamically collects data from websites which is timeliness with the focus of hot topics. Training set contains 5200 documents in 5 different classes, while testing set has 2500 documents.

(2) TREC: the dataset (http://trec.nist.gov/data.html) based on a subset of the AP newswire stories has 242,918 stories. Over 50,000 texts are selected from TREC randomly, reporting events from areas as different as the politics, finance, media, entertainment, and so forth.

(3) 20 Newsgroups Dataset (http://www.qwone.com/~jason/20Newsgroups/): this dataset is a collection of approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups. 2,0000 texts are randomly selected for the classification experiments. Of them 3,000 are multilabel texts and the rest are single-labeled.

Three sets of baseline approaches are chosen for the experiments.

(1) k-NN approach: this approach finds the nearest k neighbors in the training set. After finding the neighbors, it can be calculated how many of these neighbors belong to the nth class. Therefore, the probabilities of the test points belonging to each of the classes can be got by dividing the counts with k.

(2) Term vector model: it is an algebraic model for representing texts as vectors of identifiers, which is used in information filtering, retrieval, indexing, and relevancy rankings. VSM [22] signs the importance of topics by the term weights computed as the term frequencies.

(3) Multilabel classification approaches: MetaLabeler [23] can determine the relevant set of labels for each instance without intensive human involvement or expensive cross-validation. Two steps are involved: one is to construct the metadata; the other is to learn a metamodel. The first step can be considered as a multiclass classification problem.

The size of training dataset should be kept within a reasonable Scope. A small amount of data could affect the authority of SemGraph, while a large number of data would incur unnecessary computational cost. After class-SemGraphs have been established, unknown-class texts are studied to ensure the timeliness and the quality of the corresponding class-SemGraph, so the size of training dataset is not the bigger the better. In order to explain the method of setting the size of training dataset, ten texts are made as a set to build or update a class-SemGraph. If the number of information increment is small and the added information is of low importance, the learning process will be ceased.

In Figure 6, the size of training dataset within [70, 100] is reasonable. If a class contains a relatively larger amount of information, the size of the training set should be set as a bigger value, such as Book.

Details of the training sets and the class-SemGraphs are shown in Table 1. The number of nodes in SemGraphs is compared with the size of datasets, which is shown in Figure 7. Take Car for an example; the generated knowledge is shown in Table 2 after studying 100 texts in Car. The weights in Table 2 are the sums of weights of the same nodes in different texts, which are calculated by the algorithms mentioned before.

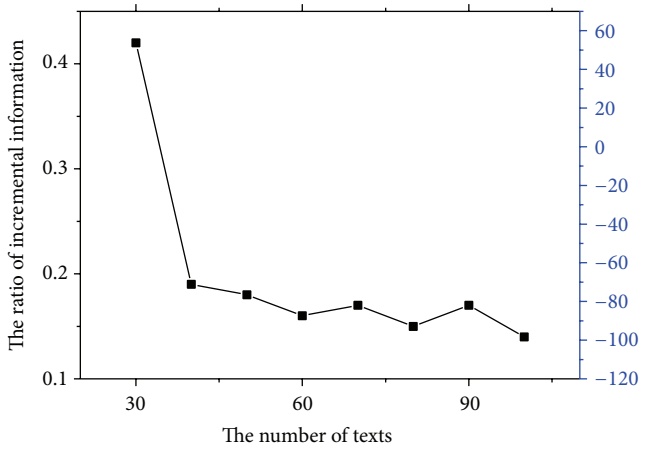

Figure 6: The contrast data between the amount of texts and incremental information ratio.

Table 1: Details of the training sets and SemGraphs.

Class	The number of the training texts	The number of nodes in class-SemGraph
Military	100	184
Car	100	169
Book	120	216
Mobile phone	100	157
Share	100	145

Adding new texts that have been categorized into the corresponding class-SemGraph can help to update it in real-time. Manually analyzing the training dataset points out that most of the texts can be classified into one or two classes (shown in Figure 8), but only the closest matching class is selected. If the algorithm maps one text to several matched classes, it will cause unnecessary troubles, because multiple matches would confuse distinctions between classes, which make text classification more difficult. For example, Car normally contains the features of the following classes: finance, energy, transportation, and environmental protection. If texts in Car are used to update the SemGraphs of finance, the two class-SemGraphs will become more similar and more difficult to be distinguished, so this paper only classifies a text as the most similar class. The final classification results are shown in Table 3.

By analyzing the experimental results, the proposed algorithm is proved to be effective, which outperforms the other algorithms and is stable to deal with different kinds of data. The result is shown in Table 4.

It can be found that the errors of the algorithm proposed are acceptable and reasonable by analyzing relationships between the wrong classified text and the class. The errors will not affect users' experiences but may indirectly influence the accuracy of class-SemGraphs. It is simple to improve this shortcoming, which is to add a judgment for filtering inappropriate texts before updating class-SemGraphs. Instead of using all the new texts to update class-SemGraphs, the improved method is to select the texts which highly

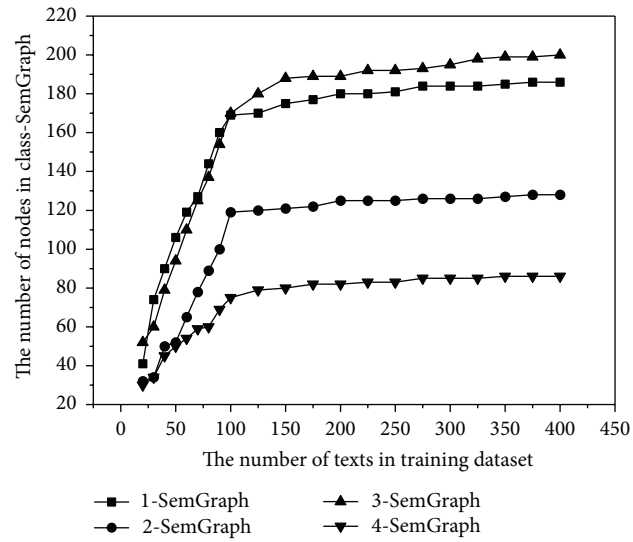

Figure 7: Data comparisons. The number of nodes in different class-SemGraphs is compared with the size of datasets.

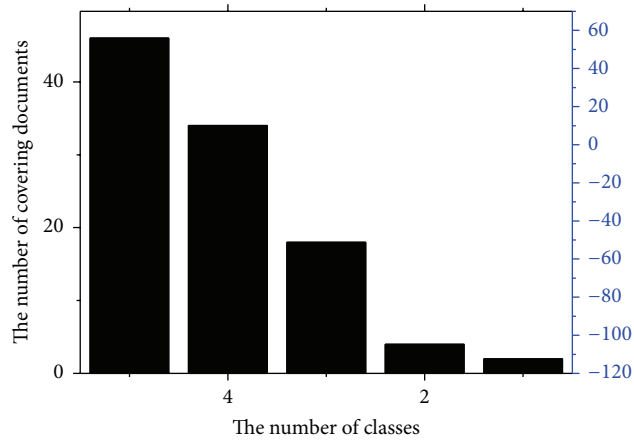

Figure 8: The numbers of classes by numbers of covering documents.

match with one certain class and have low matching degree with other classes to update class-SemGraphs. If we wish to keep class-SemGraphs entirely pure, the texts only matching one class are chosen to update the corresponding class-SemGraph. It can be found from Figure 8 that the number of texts belonging to one class is the largest, so this method is feasible.

6. Conclusion

Compared with other mainstream methods, the method proposed is simple and able to discover implicit knowledge. In addition, the algorithm is more stable in dealing with different kinds of data. After analyzing classification results, it is found that errors fall within a reasonable range and the relationship between the incorrectly classified text and the wrongly specified class makes some senses.

Table 2: Nodes in the class-SemGraph of Car (Top 50).

Number	Noun	Weight	Number	Noun	Weight	Number	Noun	Weight
1	Model	13.66	18	Vehicle	1.18	35	Science	0.69
2	Automobile	10.77	19	Car	1.16	36	Line	0.68
3	Market	3.61	20	Santana	1.14	37	Versions	0.67
4	System	3.07	21	Bloc	1.13	38	Price	0.67
5	Engine	2.93	22	Brand	1.1	39	Function	0.66
6	Volkswagen	2.52	23	Jaguar	1.02	40	Manipulate	0.66
7	Besturn	2.3	24	Luxgen	1	41	Place	0.65
8	China	2.25	25	Ford	0.99	42	Business	0.65
9	FAW	2.17	26	Technology	0.92	43	Toyota	0.65
10	Power	2.16	27	Dongfeng	0.9	44	Motive power	0.62
11	Platform	1.5	28	Turbine	0.9	45	Car body	0.6
12	Audis	1.5	29	Intelligence	0.82	46	Weight	0.58
13	Lamp	1.49	30	Benz	0.77	47	Chrysler	0.57
14	Hybrids	1.45	31	Performance	0.76	48	Company	0.55
15	BMW	1.43	32	Citroen	0.75	49	Hongqi	0.55
16	Underpan	1.34	33	Chery	0.75	50	Space	0.55
17	Consumer	1.19	34	Gasoline	0.69			

Table 3: Classification results of different classes.

	P (%)	R (%)	F-score
C-1	84.00	84.00	84.00
C-2	**88.00**	84.61	86.27
C-3	82.00	83.67	82.82
C-4	**88.00**	**93.61**	**90.71**
C-5	86.00	84.31	85.14
Ave.	85.60	86.04	85.79

Table 4: Experiment results compared with other methods.

Dataset	Method	P (%)	R (%)	F-score
SND	IStruc	75.60	76.04	**75.82**
	k-NN	45.55	64.13	58.61
	VSM	62.60	64.00	63.30
	MetaLabeler	70.54	78.46	74.29
TREC	IStruc	85.21	78.86	**81.91**
	k-NN	59.15	58.45	58.79
	VSM	65.00	60.87	62.87
	MetaLabeler	84.12	70.66	76.80
20 Newsgroups	IStruc	79.11	72.14	**75.46**
	k-NN	68.45	70.45	69.43
	VSM	54.11	65.44	59.24
	MetaLabeler	79.45	70.03	74.44

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This project is supported by National Natural Science Foundation of China (60973040); National Natural Science Foundation of China for Young Science (61300148); Key Scientific and Technological Breakthrough Program of Jilin Province (20130206051GX); China Postdoctoral Science Foundation Funded Project (2012M510879); and Scientific Frontier and Cross Project of Jilin University (201103129).

References

[1] J. Manyika, M. Chui, and B. Brown, "Big data: the next frontier for innovation, competition, and productivity," Tech. Rep., McKinsey Global Institute (MGI), 2011.

[2] G. Costa and R. Ortale, "On effective XML clustering by path commonality: an efficient and scalable algorithm," in *Proceedings of the IEEE 24th International Conference on Tools with Artificial Intelligence (ICTAI '12)*, pp. 389–396, IEEE, Athens, Greece, November 2012.

[3] P. Antonellis, C. Makris, and N. Tsirakis, "XEdge: clustering homogeneous and heterogeneous XML documents using edge summaries," in *Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC '08)*, pp. 1081–1088, March 2008.

[4] M. J. Zaki and C. C. Aggarwal, "XRules: an effective algorithm for structural classification of XML data," *Machine Learning*, vol. 62, no. 1-2, pp. 137–170, 2006.

[5] S. Tan, "An effective refinement strategy for KNN text classifier," *Expert Systems with Applications*, vol. 30, no. 2, pp. 290–298, 2006.

[6] G. Costa, G. Manco, R. Ortale, and E. Ritacco, "Hierarchical clustering of XML documents focused on structural components," *Data and Knowledge Engineering*, vol. 84, pp. 26–46, 2013.

[7] J.-B. Gao, B.-W. Zhang, and X.-H. Chen, "A WordNet-based semantic similarity measurement combining edge-counting and information content theory," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 80–88, 2015.

[8] S. Joshi, N. Agrawal, R. Krishnapuram, and S. Negi, "A bag of paths model for measuring structural similarity in Web documents," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, pp. 577–582, August 2003.

[9] K. Robles, A. Fraga, J. Morato, and J. Llorens, "Towards an ontology-based retrieval of UML class diagrams," *Information and Software Technology*, vol. 54, no. 1, pp. 72–86, 2012.

[10] H.-C. Chu, M.-Y. Chen, and Y.-M. Chen, "A semantic-based approach to content abstraction and annotation for content management," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2360–2376, 2009.

[11] D. Sánchez and D. Isern, "Automatic extraction of acronym definitions from the Web," *Applied Intelligence*, vol. 34, no. 2, pp. 311–327, 2011.

[12] J. Marés and V. Torra, "On the protection of social networks user's information," *Knowledge-Based Systems*, vol. 49, pp. 134–144, 2013.

[13] M. Batet, A. Erola, D. Sánchez, and J. Castellà-Roca, "Utility preserving query log anonymization via semantic microaggregation," *Information Sciences*, vol. 242, pp. 49–63, 2013.

[14] M. Liu, W. M. Shen, Q. Hao, and J. W. Yan, "An weighted ontology-based semantic similarity algorithm for web service," *Expert Systems with Applications*, vol. 36, no. 10, pp. 12480–12490, 2009.

[15] P. Resnik, "Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language," *Journal of Artificial Intelligence Research*, vol. 11, pp. 95–130, 1999.

[16] A. Tagarelli, "Exploring dictionary-based semantic relatedness in labeled tree data," *Information Sciences*, vol. 220, pp. 244–268, 2013.

[17] W. De Smet and M.-F. Moens, "Representations for multi-document event clustering," *Data Mining and Knowledge Discovery*, vol. 26, no. 3, pp. 533–558, 2013.

[18] Y. Guo, Z. Shao, and N. Hua, "Automatic text categorization based on content analysis with cognitive situation models," *Information Sciences*, vol. 180, no. 5, pp. 613–630, 2010.

[19] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 993–1022, 2003.

[20] Y. Xin, J. Yang, Z.-Q. Xie, and J.-P. Zhang, "An overlapping semantic community detection algorithm base on the ARTs multiple sampling models," *Expert Systems with Applications*, vol. 42, no. 7, pp. 3420–3432, 2015.

[21] T. Zesch and I. Gurevych, "Wisdom of crowds versus wisdom of linguists—measuring the semantic relatedness of words," *Natural Language Engineering*, vol. 16, no. 1, pp. 25–59, 2010.

[22] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, 1989.

[23] L. Tang, S. Rajan, and V. K. Narayanan, "Large scale multi-label classification via MetaLabeler," in *Proceedings of the 18th International World Wide Web Conference (WWW '09)*, pp. 211–220, New York, NY, USA, April 2009.