Knowledge-Based Systems 23 (2010) 302–308

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Text classification using graph mining-based feature extraction

Chuntao Jiang *, Frans Coenen, Robert Sanderson, Michele Zito

The University of Liverpool, Department of Computer Science, Ashton Building, Ashton Street, Liverpool, L69 3BX, United Kingdom

ARTICLE INFO

Article history:
Available online 22 November 2009

Keywords:
Text classification
Graph representation
Graph mining
Weighted graph mining
Feature extraction

ABSTRACT

A graph-based approach to document classification is described in this paper. The graph representation offers the advantage that it allows for a much more expressive document encoding than the more standard bag of words/phrases approach, and consequently gives an improved classification accuracy. Document sets are represented as graph sets to which a weighted graph mining algorithm is applied to extract frequent subgraphs, which are then further processed to produce feature vectors (one per document) for classification. Weighted subgraph mining is used to ensure classification effectiveness and computational efficiency; only the most significant subgraphs are extracted. The approach is validated and evaluated using several popular classification algorithms together with a real world textual data set. The results demonstrate that the approach can outperform existing text classification algorithms on some dataset. When the size of dataset increased, further processing on extracted frequent features is essential.

1. Introduction

The most common document formalisation for text classification is the *vector space* model founded on the bag of words/phrases representation. The main advantage of the vector space model is that it can readily be employed by classification algorithms. However, the bag of words/phrases representation is suited to capturing only word/phrase frequency; structural and semantic information is ignored. It has been established that structural information plays an important role in classification accuracy [14].

An alternative to the bag of words/phrases representation is a graph based representation, which intuitively possesses much more expressive power. However, this representation introduces an additional level of complexity in that the calculation of the similarity between two graphs is significantly more computationally expensive than between two vectors (see for example [16]). Some work (see for example [12]) has been done on hybrid representations to capture both structural elements (using the graph model) and significant features using the vector model. However the computational resources required to process this hybrid model are still extensive.

The computational complexity of the graph representation for text mining is the main disadvantage of the approach, which prevents the full exploitation of the expressive power that the graph representation possesses. The work described in this paper seeks to address this issue by applying weighted graph mining analysis to the problem. The intuition behind the approach is that in standard frequent subgraph mining all generated subgraphs are assumed to have equal importance. However it is clear that, at least in the context of text mining, some subgraphs are more significant than others.

The rest of this paper is organized as follows. In Section 2 a brief overview of previous work is presented. The graph representation of document sets is then discussed in Section 3. In Section 4 the weighted subgraph mining is defined. The proposed weighted graph mining algorithm, a variation of gSpan called Weighted gSpan (W-gSpan), is introduced in Section 5. A set of evaluating experiments are then presented in Section 6, followed by some concluding remarks in Section 7.

2. Related work

Much early work on document graph representations for text classification was directed at Web documents. Geibel et al. in [7] demonstrated that it is possible to classify Web documents using document structure alone; however we shall demonstrate that a much more powerful approach is to combine structure with linguistic and semantic information. For example Schenker [16] proposed a number of methods to represent Web documents as graphs so as to include the structural information of the Web documents. The typical approach is to conduct classification using some similarity-based algorithm. However, approaches that operate using graph similarity measures are computationally expensive (for example computing the "maximum common subgraph" between two graphs is a NP hard problem [5]). Hybrid representations have been introduced to resolve the computational overhead associated with pure graph representations, see for example [12]. Such hybrid representations are reported to have

* Corresponding author. Tel.: +44 0151 7954275; fax: +44 0151 7954235.
E-mail addresses: c.jiang@liv.ac.uk (C. Jiang), coenen@liv.ac.uk (F. Coenen), azaroth@liv.ac.uk (R. Sanderson), michele@liv.ac.uk (M. Zito).

0950-7051/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2009.11.010

better performance than pure graph based methods. However the computational resources required to process these hybrid model are still very high due to: (i) the extremely high number of nodes and edges, low number of edge labels and high repetition of structural node labels, encountered; and (ii) the consequent exponential complexity of the search space.

The use of graphs for representing text has a very long history in Natural Language Processing (NLP). However the work in NLP has focused on language understanding techniques such as Part Of Speech (POS) tagging, rather than text classification. Previous work [13,20] has looked at the collocation of terms and their frequencies as graphs, rather than the linguistic structure of the sentence. One other study [6] has represented linguistic information as well as word order in a graph for text classification, however the work was limited to very small texts of between 8 and 13 tokens such as the titles of works. As such, we adopt the usage of linguistic information, structure and semantics in a graph for text classification at a full text scale. In order to achieve this scale of processing, the use of frequent subgraph mining is essential.

Frequent subgraph (and sub-tree) mining, using various approaches, has been extensively studied [9,10,22,8,2]. However, the main bottleneck is the number of unnecessary candidate frequent subgraphs generated. A substantial amount of work has been undertaken focusing on developing efficient graph mining algorithms using elegant search strategies, data structures or their combinations. Some authors have suggested the use of constraint based frequent subgraph mining to remove unwanted patterns. The weighted subgraph mining approach advocated in this paper integrates the weight constraints into the frequent subgraph mining process to reduce the search space by generating only the most significant (interesting) patterns.

The frequent subgraph mining approach described in this paper is also influenced by work on weighted pattern mining, especially Weighted Association Rules Mining (WARM), see for example the work of [19,17,23–25]. A significant issue in WARM is that the "Downward Closure" (DC) property of items sets, on which many ARM algorithms are based, no longer holds. One solution (for example [19]) is to handle the weights as a post-processing step after mining frequent itemsets, however the weights are then not integrated into the ARM process. Tao et al. [17] proposed a model of weighted support, which satisfies a weighted DC property. Yun et al. [23–25] introduced a series of concepts such as "weight range", "weight confidence", and "support confidence" for WARM in order to maintain the DC property and push the weight constraint deeply into the mining process. Although the ideas espoused by WARM cannot be directly applied to weighted frequent subgraph mining; the research described here is, at least in part, influenced by this body of work.

3. Graph representation of text data

The graph representation advocated in this paper is described in this section. The representation serves to capture a range documents aspects: (i) word stem, (ii) word Part Of Speech (POS), (iii) word order, (iv) word hypernyms, (v) sentence structure, (vi) sentence division and (vii) sentence order. There are four different types of nodes in the graph representation:

1. *Structural*: Nodes that represent sentences (S) and their internal structures of noun (NP), verb (VP) and prepositional phrases (PP). (Represented by triangles in Fig. 1.)
2. *Part of Speech*: Nodes that represent the POS of a word, (e.g. DT, JJ, and NN). (Circles.)
3. *Token*: Nodes that represent the actual word tokens in the text. (Rectangles.)
4. *Semantic*: Nodes that represent additional information about the word such as its linguistic stem and other broader concepts. (Ovals)

Note that each node has a unique identifier and a label. There are also five types of edge in the graph:

1. *hasChild*: Edges which record the structure of the text such as a sentence having a noun phrase and a verb phrase or a noun phrase containing an adjective. (Unlabeled in Fig. 1 for reasons of space.)
2. *isToken*: Edges which link the part of speech of a token to the token itself.
3. *next*: Edges which record the order of the words and sentences in the text.
4. *stem*: Edges which link to the linguistic stem of the word.
5. *hyp*: Edges which link to a broader concept.

An example of these node and edge types is depicted in Fig. 1, using the first 6 words in a well known English sentence. Employing the above graph representation each sentence in each text is encoded and linked together with "next" edges to form one graph per text. Content based weightings were then attached to each node in the graph. The Structural elements, being intuitively unimportant to classification, were given a static low weight of 1. The Part of Speech nodes were given a static weight of 10, Token nodes were weighted according to their frequency in the dataset using the $TF \cdot IDF$ method. Stems were half the value of the Token and Hypernyms one quarter the value.

4. Weighted frequent subgraphs

In this section the weighted subgraph mining problem is formally defined. As with standard transaction graph mining approaches [9,10,1,11] we commence with a set of *transaction graphs* $D = \{G_1, G_2, \ldots, G_n\}$ and a function $\tau(g, G)$ for arbitrary graphs g and G. $\tau(g, G) = 1$ (resp. 0), if g is isomorphic to a subgraph in G.

Definition 1. The support count of a graph (pattern) g with respect to a database $D = \{G_1, G_2, \ldots, G_n\}$, is the expression $sco(g) = \sum_{i=1}^{i=n} \tau(g, G_i)$. The support of g with respect to D, $sup(g)$, is the ratio of the support count over the size of the dataset D. Then:

$$sup(g) = \frac{sco(g)}{n}. \quad (1)$$

It should be remarked that $sco(g)$ and $sup(g)$, like most terms defined in this section depend on the dataset D. To avoid cluttering notations, such dependence will always be left implicit.

Definition 2. Given a graph g, if $sup(g)$ is greater than or equal to some user defined minimum threshold θ, then g is said to be frequent (in D). The frequent subgraph mining problem is to find all the frequent subgraphs in the database D.

Since the purpose of this paper is to study weighted graph mining in the remainder of this section we define this concept precisely. From now on we assume that graphs come with weights associated with either their vertices or their edges. Let W be a function assigning a weight to any graph g in terms of the given weights for its vertices (resp. edges). In our work, in particular, W will always be a sum of the vertex (resp. edge) weights, but the definitions in this section hold in a more general setting.

Definition 3. Given a graph g with the weight $W(g)$, the weighted support of g with respect to D, $wsup(g)$, is:

$$wsup(g) = W(g) \times sup(g). \quad (2)$$

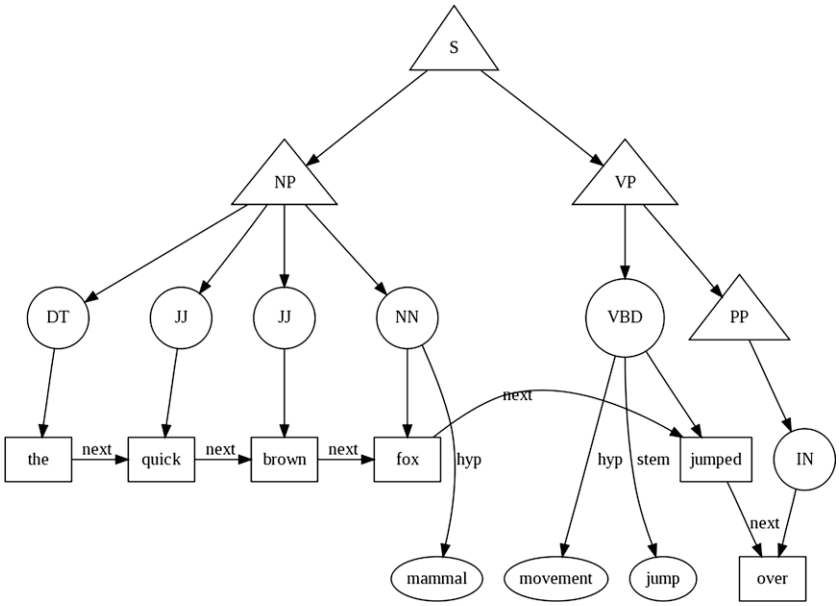

Fig. 1. Graph-based text representation example.

Definition 4. A graph g is said to be weighted frequent if and only if its weighted support is greater than or equal to a given minimum support threshold (minwsup),

$$wsup(g) \geqslant minwsup. \tag{3}$$

From Eqs. (1)–(3), a graph g is weighted frequent if its support count satisfies:

$$sco(g) \geqslant \frac{minwsup \times n}{W(g)}. \tag{4}$$

Note that $sco(g)$ is always an integer. Hence we may define

$$sbound(g) = \left\lceil \frac{minwsup \times n}{W(g)} \right\rceil \tag{5}$$

and we have

$$sco(g) \geqslant sbound(g). \tag{6}$$

5. Weighted gSpan

The operation of the proposed Weighted subgraph mining Algorithm (W-gSpan) is described in this section. The section commences (Section 5.1) with a discussion of support-bound candidate subgraph pruning. This is followed in Section 5.2 by a description of a number of different weighting mechanisms that are used in this study. Section 5.3 then gives the pseudo code of pruning algorithm and briefly describes how W-gSpan is integrated into the classification process.

5.1. Support bound based pruning

Use of the DC property in any frequent set mining algorithm can greatly reduce the search space. However, in the context of weighted frequent set mining the DC property no longer holds. The W-gSpan algorithm therefore makes used of an alternative concept to prune non-interesting candidate subgraphs early on in the generation process.

Let the maximum possible size of a subgraph be mL and the weight for a subgraph be defined as the sum of vertex weights (similar definitions may be given if the graph is edge-weighted). Given a k-pattern g_k with weights $\{\omega_1, \omega_2, \ldots, \omega_k\}$, any future n-pattern containing g_k is denoted by g_n, where $k < n \leqslant mL$. For the additional $(n-k)$ vertices, if the upper bounds of the weights are estimated as $\omega_{a_{k+1}}, \omega_{a_{k+2}}, \ldots, \omega_{a_n}$, then the upper bound of the weight of the n-pattern g_n is given by:

$$wbound_n(g_k) = \sum_{i=1}^{k} \omega_i + \sum_{i=k+1}^{n} \omega_{a_i} \tag{7}$$

We may then define a lower bound of the support count of a k-pattern included in g_n as

$$\underline{sbound}_n(g_k) = \left\lceil \frac{minwsup \times n}{wbound_n(g_k)} \right\rceil \tag{8}$$

of course the definition can be extended to $n = k$ by setting $\underline{sbound}_k(g_k) = sbound(g_k)$ as defined in Eq. (5).

Definition 5. A k-subgraph g_k is *workable* if $sco(g_k) \geqslant \underline{sbound}_n(g_k)$ for some n with $k \leqslant n \leqslant mL$, and *unworkable* if $sco(g_k) < \underline{sbound}_n(g_k)$ for all n, with $k \leqslant n \leqslant mL$.

Lemma 1. *If a subgraph g_k is workable then it is possible for g_k to be a subgraph of some weighted frequent n-subgraph. On the contrary, if a subgraph g_k is not workable, then g_k has no possibility of being a subgraph of any weighted frequent n-subgraph.*

Proof. Let n be given with $k \leqslant n \leqslant mL$. If $sco(g_k) \geqslant \underline{sbound}_n(g_k)$, then due to $sco(g_k) \geqslant sco(g_n)$, it is possible that $sco(g_n) \geqslant sbound(g_n)$. So pattern g_n has a chance to be weighted frequent in the future. On the other hand, if $sco(g_k) < \underline{sbound}_n(g_k)$, then due to $sco(g_k) \geqslant sco(g_n)$, $sco(g_n) < sbound(g_n)$. So pattern g_n will not be weighted frequent in the future. □

The Weighted gSpan algorithm will then use a simple condition to decide whether or not to prune a particular k-pattern (in what follows mL is the maximum length of a pattern):

> if $sco(g_k) \geqslant sbound(g_k)$, g_k is workable; otherwise we compute $\underline{sbound}_{mL}(g_k)$ (this gives a lower bound on $\underline{sbound}_n(g_k)$), if $sco(g_k) \geqslant \underline{sbound}_{mL}(g_k)$, then g_k is workable, else g_k is unworkable and pruned.

5.2. Weight calculation

Given the notion of a weighted bound of a subgraph, as defined above, methods for calculating the weighting for a given subgraph are required. We can identify three approaches for determining subgraph weightings: (i) **structure based**, (ii) **content based** and (iii) **structure and content** based. The distinction between the two is that the structure base weighting approach does not require any advanced knowledge of the potential significance of subgraphs. Each approach is discussed in more detail below.

5.2.1. Structure based weight calculation

In the structure base weighting approach weightings are derived purely from the "structure" of subgraphs. The approach advocated here is based on the frequency counts of individual nodes and edges per graph in the graph set. Using these frequency counts we adopt Pearson's Correlation Coefficient [15], PCC, to measure the weight of the edge (considering the nodes making up a 1-edge subgraph as two variables). Thus for two nodes A and B, let the number of occurrences of A equal ϕ_A, the number of occurrences of B equal ϕ_B and the number of co-occurrences of A and B equal ϕ_{AB}; and let the total number of transaction graphs within the dataset be equal to n. The support values will then be $sup(A) = \phi_A/n$, $sup(B) = \phi_B/n$, and $sup(A,B) = \phi_{AB}/n$. Using PCC the edge weight (ω_{pcc}) can be derived as follows.

$$\omega_{pcc} = \frac{sup(A,B) - sup(A)sup(B)}{\sqrt{sup(A)sup(B)(1-sup(A))(1-sup(B))}} \tag{9}$$

Many other measures of association exist, such as the Chi Squared, cosine or Jaccard measure, that could equally well be used to determine edge weighting in a structured based context.

5.2.2. Content based weight calculation

In the content based weighting approach advanced knowledge of the nature of the input set is utilised. The nature of the advanced knowledge can take two forms: (i) weights that have been predefined (by for example a domain expert), or (ii) class labels associated with individual graph records (documents).

In the first case user supplied weightings can be attached directly to either nodes or edges. Thus given a set of user defined node weights $\omega_1, \ldots, \omega_n$, the weighting for a subgraph can be calculated by $\sum_{n_i \in g} \omega_i$. A similar calculation can be used in the event of user supplied edge weights. We later refer to this mechanism as the "Node Weight" method.

Alternatively we can calculate edge weights, given user defined node weights, as follows: if the nodes connecting edge e_i are a with weight w_a and b with w_b; the probability of a's occurrences is ρ_a, the probability of b's occurrences is ρ_b and the probability of edge e_i's occurrences is $\rho(a,b)$. The mutual information metric between a and b can then be defined as $mu(a,b) = \rho(a,b)\log_2(\rho(a,b)/\rho_a/\rho_b)$. The weight for edge e_i can then be calculated as:

$$\omega_{e_i} = \left(\frac{2 \times w_a \times w_b}{w_a + w_b}\right) \times mu(a,b). \tag{10}$$

The weight for the subgraph is calculated in the same manner as before. We refer to this mechanism as the "Mu" method.

Alternatively knowledge of the class label can be used to determine the weighting of a given subgraph. There are a number of *feature selection* techniques that can be utilised for this purpose, examples include Information Gain (IG), mutual information(MI), and χ^2 testing. For the work described here the χ^2 statistic was adopted to apply weightings to subgraphs according to their association with a given class label. Using the two-way contingency table of an edge e and a graph's class label y_c, let a denote the number of times e and y_c co-occur, b denote the number of times the e occurs without y_c, c denote the number of times y_c occurs without e, d denote the number of times neither e nor y_c occurs, and n is the total number of transaction graphs. The edge-goodness measure is then defined to be:

$$\chi^2(e,y_c) = \frac{n(ad-cb)}{(a+c)(b+d)(a+b)(c+d)}. \tag{11}$$

The χ^2 statistic has a value of zero if edge e and class y_c are independent. For each class y_c, we compute the χ^2 statistic between each edge and that category, and then calculated the average value of χ^2 statistic for each edge. Let $c = \{c_1, c_2, \ldots, c_m\}$ denote the set of categories for the transaction graphs dataset, $P_r(y_c)$ denotes the probability of y_c, then:

$$\chi^2_{avg}(e) = \sum_{c=1}^{m} P_r(y_c)\chi^2(e,y_c). \tag{12}$$

After estimating edge weights for each generated subgraph, the actual significance of the subgraph is calculated in the same manner as before. We refer to this mechanism as the "Chi Squared" method.

5.2.3. Combined content and structure based weight calculation

It is possible to combine the two approaches, content and structure based weight calculation. For example given a user defined weight for node n_i of w_{n_i}, then the probability of n_i's occurrences is ρ, and the entropy for node n_i is $entropy(n_i) = -\rho\log(\rho) - (1-\rho)\log(1-\rho)$. If we also make use of the "degree" (the number of edges incident to the node) of n_i the weight for n_i can be calculated as:

$$\omega_{n_i} = w_{n_i} \times entropy(n_i) \times degree(n_i) \tag{13}$$

Thus, we refer to this mechanism as the "Entropy" method.

Algorithm 1. [subgraph-mining(GS, s, c, F)]

Require: Input: c = DFS code, GS = graph database, s = support;
Ensure: Output: F = weighted frequent subgraph set;
1. $G \leftarrow$ a set of candidate subgraphs;
2. **if** $c \neq min(c)$ **then**
3. **return**
4. **end if**
5. Insert c into F;
6. $G \leftarrow \emptyset$;
7. Scan GS once, and find every edge e that c can be right-most extended, and save $c \cup e$ into G;
8. Sort G in DFS lexicographic order;
9. **for all** $g_k \in G$
10. **if** $sco(g_k) \geqslant sbound(g_k)$
11. subgraph-mining(GS, s, c, F);
12. **else if** $sco(g_k) \geqslant min(\underline{sbound}_n(g_k))$, where $g_k \subset g_n$
13. subgraph-mining(GS, s, c, F);
14. **else**
15. $G \leftarrow G - \{g_k\}$;
16. **end if**
17. **end for**
18. **return**

5.3. The Weighted gSpan Algorithm (W-gSpan)

The above weighting considerations were built into a variation of the well known gSpan frequent subgraph mining algorithm [22], Weighted gSpan (W-gSpan). However, the proposed weighing framework can equally well be applied to other frequent subgraph (or sub-tree) mining algorithms. The pseudo code for the pruning algorithm employed in W-gSpan is given in Algorithm1.

After the W-gSpan algorithm is applied to identify weighted frequent subgraphs, these subgraphs are then used to generate a set of binary feature vectors (one per document). A standard classifier generator can then be employed using such vectors.

6. Experiments and results

In order to evaluate the performance of the proposed graph based text classification method experiments were conducted to:

- Investigate the performance of W-gSpan, in terms of execution time and number of frequent subgraphs detected.
- Investigate the overall performance of the graph based classification process for text classification.

Note that the experiments were all run on a 1.86 GHZ Intel Core 2 PC with 2 GB main memory.

6.1. Description of text data set

The experimental data consisted of three sets of documents (D1–D3) split evenly between two classes. The documents were extracted from the Medline dataset by their Medical Subject Heading (MeSH) fields, so that a two class (“polymerase chain reaction” and “magnetic resonance imaging”) set was produced. The text was divided into sentences using a regular expression based tokenizer and then each sentence was POS tagged using Tsuruoaka and Tsujii's “geniatagger” [18], producing a sequence of “word/POS” tokens plus the lemma (stemmed form) of each word. This tagged output was then fed into a structural parser which produces a tree with noun, verb and prepositional phrases. The nouns and verbs are then “looked up” in the WordNet thesaurus and up to five broader terms added into the graph. The properties of the (graph) data are given in Table 1.

Table 1
Graph data description.

Text dataset	D1	D2	D3
No. of graphs	200	400	1000
Maximal edge count	3002	2917	4047
Average edge count	1141	1131	1135
Distinct node label count	10,069	16,456	26,540

6.2. Performance of W-gSpan

The performance of the W-gSpan algorithm was evaluated using the four different weighting methods introduced in Section 5.2 above:

- Pearson Correlation Coefficient (`pcc-w`) for structure based weighting.
- Node Weight (`node-w`) for content based node weighting (Edge weighting would operate in a similar manner).
- Mutual information (`mu-w`) for content based edge weighting.
- Chi Square (`chs-w`) for content based class label discrimination weighting.
- Node entropy (`entro-w`) for combined structure and content based node weighting.

Experiments were also conducted with no weighting, but this was found to be extremely inefficient with poor outcomes, and thus are not discussed further in this evaluation.

The results of the performance experiments are presented in Fig. 2. The runtime values corresponding to different minimum support thresholds are presented in Fig. 2a. The number of identified frequent subgraphs (features), corresponding to a range of minimum support thresholds, is presented Fig. 2b and c. There

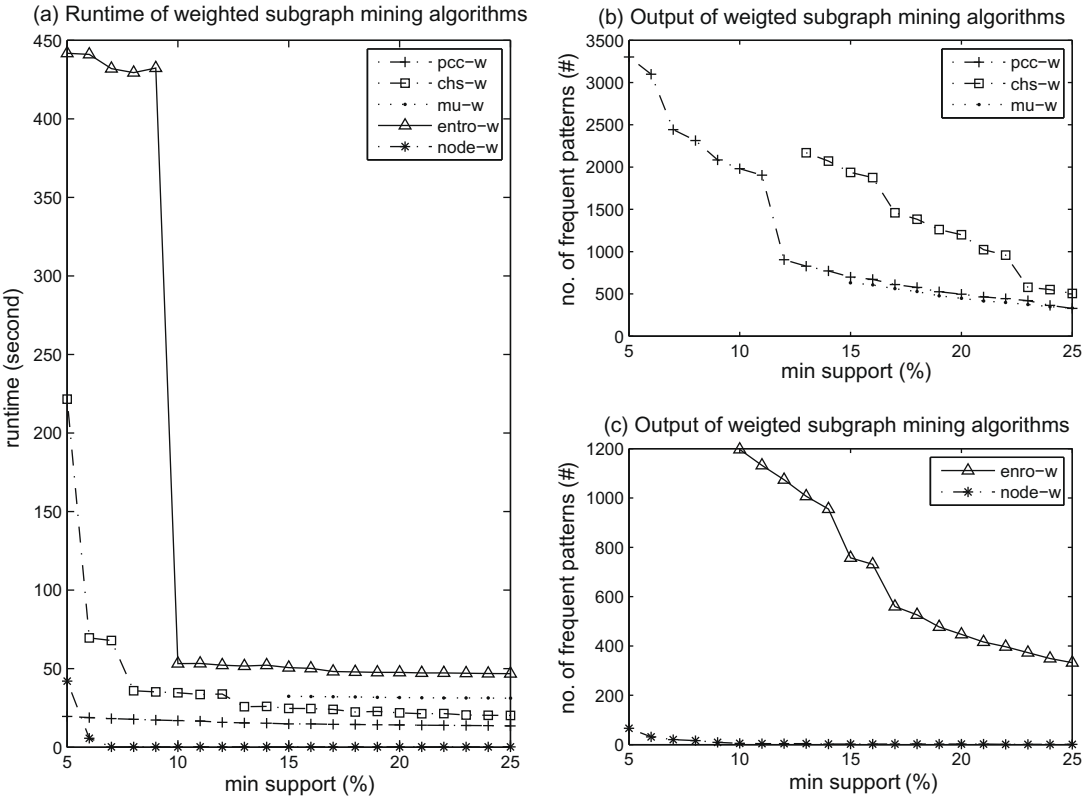

Fig. 2. Performance of weighted frequent subgraph mining on D1 dataset.

Table 2
Classification accuracy by different weighting methods on D1 dataset.

Classifier	Method	Support threshold (%)										
		15	16	17	18	19	20	21	22	23	24	25
NBC	pcc-weight	96.5	96	94.5	95	94	93	92	93	91.5	91.5	91.5
	chs-weight	91.5	91.5	89	87.5	86.5	86.5	90	91	90.5	92	91
	mu-weight	97	96.5	94.5	94	94	93	92	93	91.5	92	91.5
	entro-weight	76.5	75	95	95	94	92.5	92	92.5	92	93	92.5
	node-weight	80.5	80.5	80.5	80.5	80.5	80.5	80.5	80.5	80.5	80.5	80.5
TFPC	pcc-weight	91.5	91	88.5	89.5	86.5	84.5	83.5	84	84	84	84
	chs-weight	90.5	90	87.5	88.5	85.5	83.5	82.5	83	83.5	83	83
	mu-weight	92	91.5	89	90	87	85	84	84.5	84.5	84	84
	entro-weight	92	91.5	89	90	87	85	84	84.5	84.5	84	84
	node-weight	54	54	54	54	54	54	54	80.5	80.5	80.5	80.5
C4.5	pcc-weight	88.5	88	89.5	89	91	86.5	86.5	86.5	87	89	88.5
	chs-weight	87.5	87	89.5	89	91	86	86.5	86.5	87.5	89.5	89.5
	mu-weight	88.5	88	89.5	89	91	87	87	87	87	89	88.5
	entro-weight	88.5	88	89.5	89	91	87	87	87	87	89	88.5
	node-weight	80.5	80.5	80.5	80.5	80.5	80.5	80.5	80.5	80.5	80.5	80.5

is, naturally, a correlation between support threshold and the number of identified subgraphs: as the support threshold is increased, the number of identified subgraphs decreases. There is also a natural correlation between runtime and the number of identified frequent subgraphs: runtime increases with the number of identified frequent subgraphs. From Fig. 2a it can be observed that Mu weighting and node weighting seem to work well in terms of run time efficiency, however node weighting finds very few frequent subgraphs. The pcc weighting is the most effective in terms of computational efficiency, and works well in terms of number of features generated. Entropy weighting suddenly increases the runtime when the threshold is below 10%.

6.3. Classification accuracy comparison

Three different classifier generator paradigms were use to evaluate the graph-based text classification process: (i) a classification association rule miner, TFPC [3,4], (ii) a Naive Bayes Classifier (NBC) [21], and (iii) a decision tree classifier, C4.5 [21]. Table 2 shows the accuracy figures obtained using a range of support threshold values (for the generation of frequent subgraphs), for the three classification paradigms (with 10 folds cross validation) and using the five different weighting strategies. Experiments conducted with no weightings at all (on D1–D3 datasets) produced very poor results indicating, beyond doubt, that the proposed weighted graph mining approach provides genuine benefits.

Using no weighting on D1 dataset, it was not possible to obtain results with a support threshold below 85%. When comparing the different weighting schemes, pcc produced the best overall accuracy. Using a standard 'bag of words' approach with TFPC gave a best accuracy of 89%.

If the three classifier generators are compared, NBC performs significantly better than the other two, however C4.5 did not work well with any of the permutations of weighting and support threshold. If the three classifiers are applied on D2 and D3, the classification performance degrades. For example, using PCC weighting with support 10%, the accuracy of NBC on D2 is 76.5% and the accuracy of NBC on D3 is 72.3%. In order to get better accuracies, further processing on extracted frequent features is indispensable and how to model text data as more efficient graphs with less nodes and edges is also crucial.

7. Conclusion

An approach to text classification using a graph based representation has been described. The graph representation of text allows both the structure and content of documents to be represented. Key constructs to support text classification can then be identified using frequent subgraph mining. The disadvantage of standard frequent subgraph mining is that it is computationally expensive, to the extent that any potential advantage of the graph representation of text cannot be realised. To overcome this disadvantage a weighted subgraph mining mechanism is proposed, W-gSpan. In effect W-gSpan selects the most significant constructs from the graph representation and uses these constructs as input for classification. Experimental evaluation demonstrates that the technique works well, significantly out-performing the unweighted approach in every case. A number of different weighting schemes were considered coupled with three different categories of classifier generator. In terms of the generated classification accuracy pcc-weighting outperformed the other proposed weighting mechanisms. PCC-weighting also worked well in terms of computational efficiency and therefore represents the best overall weighting strategies.

References

[1] C.H. Cai, A.W. Fu, C.H. Cheng, W.W. Kwong, Mining association rules with weighted items, in: Proceedings of International Database Engineering and Applications Symposium, August 1998.
[2] Y. Chi, S. Nijssen, R. Muntz, J. Kok, Frequent subgree mining an overview, In Fundamenta Informaticae, Special Issue on Graph and Tree Mining 66 (1–2) (2005)161–198.
[3] F. Coenen, The LUCS-KDD TFPC Classification Association Rule Mining Algorithm, Dept. of Computer Science, The University of Liverpool, UK, 2004, <http://www.csc.liv.ac.uk/frans/KDD/Software/Apriori_TFPC/aprioriTFPC.html>.
[4] F. Coenen, P. Leng, Obtaining best parameter values for accurate classification, in: Proceedings of International Conference on Data Mining, 2005, pp. 597–600.
[5] M.R. Garey, D.S. Johnson, Computers and Intractability – A Guide to the Theory of NP-Completeness, WH Freeman and Company, New York, 1979.
[6] K.R. Gee, D.J. Cook, Text classification using graph-encoded linguistic elements, in: FLAIRS Conference, 2005, pp. 487–492.
[7] P. Geibel, U. Krumnack, O. Pustylnikow, A. Mehler, et al., Structure-sensitive learning of text types, in: AI 2007: Advances in Artificial Intelligence, vol. 4830, 2007, pp. 642–646..
[8] J. Huan, W. Wang, J. Prins, Efficient mining of frequent subgraph in the presence of isomorphism, in: Proceedings of the 2003 International Conference on Data Mining, 2003.
[9] A. Inokuci, T. Washio, H. Motoda, An apriori-based algorithm for mining frequent substructures from graph data, in: Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases, 2000.
[10] M. Kuramochi, G. Karypis, Frequent subgraph discovery, in: Proceedings of 2001 IEEE International Conference on Data Mining, 2001.
[11] S.D. Lee, H.C. Park, Mining weighted frequent patterns from path traversals on weighted graph, IJCSNS International Journal of Computer Science and Network Security 7 (4) (2007).

[12] A. Markov, M. Last, Efficient graph-based representation of web documents, in: Proceedings of the Third International Workshop on Mining Graphs, Trees and Sequences, Porto Portugal, 2005, pp. 52–62.
[13] A. Markov, M. Last, A. Kandel, Fast categorization of Web documents represented by graphs, Advances in Web Mining and Web Usage Analysis 4811 (2007) 56–71.
[14] D. Mukund, M. Kuramochi, G. Karypis, Frequent sub-structure based approaches for classifying chemical compounds, in: Proceedings of the Third IEEE International Conference on Data Mining, 2003.
[15] H.T. Reynolds, The Analysis of Cross-Classifications, The Free Press, New York, 1977.
[16] A. Schenker, Graph Theorectic Techniques for Web Content Mining, Ph.D. Thesis, University of South Florida, 2003.
[17] F. Tao, F. Murtagh, M. Farid, Weighted association rule mining using weighted support and significance framework, in: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, USA, August 2003.
[18] Y. Tsuruoka, J. Tsujii, Bidirectional inference with the easiest-first strategy for tagging sequence data, in: Proceedings of HLT/EMNLP, 2005, pp. 467–474.
[19] W. Wang, J. Yang, P.S. Yu, Efficient mining of weighted association rules (WAR), in: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, USA, August 2000.
[20] W. Wang, D.B. Do, X. Lin, Term graph model for text classification, Advanced Data Mining and Applications (2005) 9–30.
[21] Ian H. Witten, Eibe Frank, Data Mining: Practical Machine Learning Tools and Techniques, second ed., Morgan Kaufmann, San Francisco, 2005.
[22] X. Yan, J. Han, gSpan: graph-based substructure pattern mining, in: Proceedings of 2002 International Conference on Data Mining, 2002.
[23] U. Yun, J.J Leggett, WFIM: weighted frequent itemset mining with a weight range and a minimum weight, in: Proceedings of the Fifth SIAM International Conference on Data Mining, 2005, pp. 636–640.
[24] U.Yun, J.J. Leggett, WIP: mining weighted interesting patterns with a strong weight and/or support affinity, in: Proceedings of the Sixth SIAM International Conference on Data Mining, 2006.
[25] U. Yun, WIS: weighted interesting sequential pattern mining with a similar level of support and/or weight, ETRI Journal 29 (3) (2007) 336–352.